The Dissertation Committee for Shalini Ghosh certifies that this is the approved version of the following dissertation:

Reducing Power Consumption During Online and Offline Testing

Committee:

Nur A. Touba, Supervisor

Margarida F. Jacome

Anthony P. Ambler

Lizy K. John

Eric MacDonald

Reducing Power Consumption During Online and Offline Testing

by

Shalini Ghosh, B.Sc. (Hons.), M.S.

Dissertation

Presented to the Faculty of the Graduate School of
the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2005

Dedicated to my beloved family

Acknowledgments

I would like to thank my advisor Prof. Nur A. Touba for his continuous support and guidance during the course of my PhD. I have always admired his clarity of thought and command over the subject and understanding. The high standards he sets for research are an inspiration for me. I would like to thank Prof. Margarida F. Jacome, Prof. Anthony P. Ambler, Prof. Lizy K. John and Prof. Eric MacDonald for agreeing to be on my PhD committee and for reviewing my dissertation.

I could not have done my PhD without the loving support and help of my husband Sugato Basu. Not only was he a great collaborator to work with, he gave me mental support during the difficult times. His confidence in me helped me believe in myself and I would like to thank him for being there for me always.

I would like to thank all my friends at Austin for making my stay in Austin a memorable one. I would especially like to thank Sandip Ray for the long discussions that we had and for giving me the love of a brother.

I would like to thank my parents-in-law Samir Basu and Nandita Basu and all my other relatives for their support. Last but not the least, I would like to thank my father Ashok Ghosh, my mother Rajyasri Ghosh and my brother Rajiv Ghosh for their unflinching love and support. They helped me at each and every step in realizing my dream and it is to them that I dedicate this dissertation. My parents' lives are a true source of inspiration for me and I am what I am because of them.

May 2005

Reducing Power Consumption During Online and Offline Testing

Publication No. _____

Shalini Ghosh, Ph.D.

The University of Texas at Austin, 2005

Supervisor: Nur A. Touba

This dissertation proposes techniques for power reduction in online and offline circuit testing. Power management is critical in both these domains, since high power dissipation can drive up production cost and even cause errors.

The first part of the dissertation focuses on power reduction for online testing with concurrent error detection capabilities, where errors in the operation of the circuit are detected (and possibly corrected) at normal operational run-time. In online testing, power dissipation has lately become a first-order design criterion due to the significant hardware overhead for detecting/correcting errors and ensuring system reliability. Two problems are addressed, namely reducing power in concurrent error detection for (1) error correcting codes for memory checker, and (2) synthesis of parity prediction circuits.

The next part of the dissertation discusses power reduction for offline testing. With the advent of high-performance and low-power devices, the power consumed during circuit testing has become a critical issue since the power dissipated in a circuit during the testing phase can be much larger than the power consumed during normal operation. Techniques are presented for reducing power in two popular methods of offline circuit testing: (1) scan testing, and (2) built-in self-test.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The topic of this dissertation is power reduction for both online testing and offline testing of circuits. Power reduction is critical in both online testing scenarios like systems with concurrent error detection (CED) and offline testing environments like built-in self-test (BIST) for embedded systems, since high power dissipation necessitates expensive packaging of the chip to remove the excess heat and thereby drives up production cost. If power dissipation is not properly controlled, it can also cause errors, e.g., high peak power can cause ground bounce or V_{dd} drop, causing a memory cell to lose state and hence make a test fail erroneously on a good part.

Online testing: In online testing systems with concurrent error detection, errors in the operation of the circuit are detected (and possibly corrected) at normal operational run-time. Fig. 1.1 shows a typical online testing system with concurrent error detection. The logic circuit takes inputs and generates outputs, while the checker circuit generates some check bits according to a selected error control code. The error syndrome generator looks at the circuit output and the check bits to detect (and if possible correct) errors in the circuit.

Designs with on-chip hardware for error detection and correction are important in mission-critical applications where dependability of the system and integrity of the data are critical. Recently, the increase in density and reduction in size of integrated circuits, along with the lowering of voltage levels and reduction of noise margins, has made systems more susceptible to transient and intermittent faults. As a result, there has been lately an increased necessity for the design of systems with online testing.

Circuits that implement error-correcting codes for online testing based on concurrent error detection have different amounts of power consumption, depending on the hardware design. In online testing, power reduction has lately become a first-order design criterion due to the significant hardware overhead for system reliability, which dissipates significant power. The focus in this case is on power reduction in the error detection and correction circuitry while the circuit is performing its usual functions.

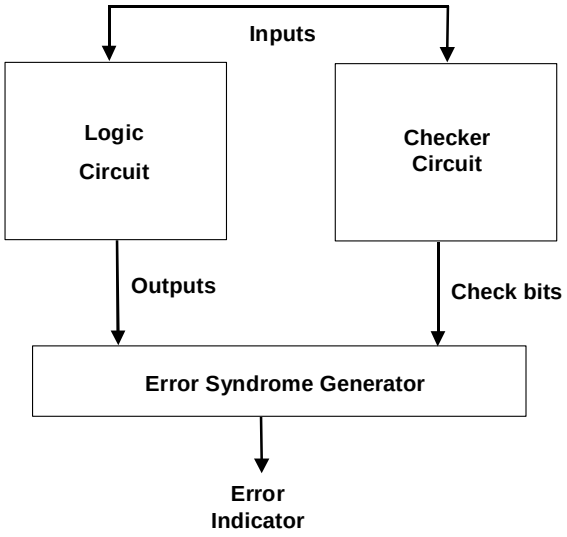

Figure 1.1: Concurrent Error Detection

Offline testing: In offline testing, there is a specific testing mode of the circuit, which is different from the normal operation mode. Fig. 1.2 shows a typical circuit with offline testing. During testing mode, a pattern generator, which can be either an external device (tester) or built-in circuitry on the chip (built-in self-test), is used to generate test patterns that are given as input to the circuit under

test (CUT). The output response of the CUT is then compared to stored correct responses to detect possible faults in the circuit. Typically, offline testing for potential faults is an expensive part of the chip production cost.

With the advent of high-performance and low-power devices, the power consumed during testing has become a critical issue. While studying offline testing, we focus on reducing power in the test mode. It has been observed that the power dissipated in a circuit during the testing phase in test mode can be much larger than the power consumed during normal operations, due to a number of causes, e.g., increased switching activity due to less correlated input vector application in test mode, use of parallel testing to reduce test application time, etc. This has made power optimization a very important problem in offline testing.

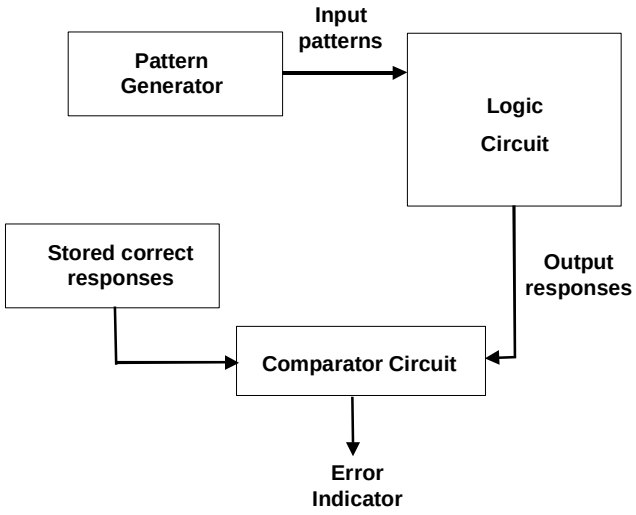

Figure 1.2: Offline Testing

1.1 Overview of Related Work

There has been recent work in power reduction for concurrent error detection circuits, including input ordering in symmetric checkers for power reduction by considering signal probability on a level-by-level basis [Favalli 97] or by analyzing spatial correlation among signals in parity trees and Berger code checkers [Mohanram 02], and power reduction in fault-tolerant buses using dual-rail codes instead of Hamming codes as the SEC code [Rossi 02, 03].

Various techniques have been used for reduction of power in offline testing. In *external* testing, Wang, *et al*., proposed low-power ATPG algorithms that reduce power along with meeting coverage objectives [Wang 97B], while Dabholkar, *et al*., proposed scan-chain re-ordering techniques to reduce transition in the circuit elements [Dabholkar 98]. Huang, *et al*., developed an algorithm for controlling input patterns with vector or latch ordering techniques for full-scan circuits [Huang 99]. Other popular power reduction techniques for external testing include scan chain transformation using separate scan paths or interleaving scan architecture [Whetzel 00], test vector compaction by merging suitable test cube pairs to reduce average and peak power [Sankaralingam 01], clock-scheme modification to selectively disable clocks [Pouya 00], or using gated clocks to reduce clock speed during scans shifting without increasing testing time [Bonhomme 01].

In BIST, Zorian, *et al*., developed test-scheduling algorithms using a distributed BIST control scheme [Zorian 93], while Hertwig, *et al*., proposed a scheme of toggle suppression for scan-based BIST by modifying scan cells [Hertwig 98]. In LFSR-based BIST, methods were developed for low power test pattern generation using modified LFSRs [Wang 97A] or by tuning the LFSR to select special polynomials that cause less power dissipation [Girard 99A]. Methods in scan-based BIST include toggle suppression by modifying scan cells

[Hertwig 98], and filtering non-detecting patterns [Girard 99C]. Many other methods for power reduction have also been proposed, e.g., circuit partitioning to split the test session and reduce average power dissipation, etc.

1.2 Overview of Dissertation

In Chapter 2, a method is proposed for reducing power consumption while performing online error detection and correction in memory using SEC-DED checker circuits. Error correcting codes (ECCs) are commonly used in memories to protect against soft errors and thereby enhance system reliability and data integrity. *Single-error-correcting* and *double-error-detecting* (SEC-DED) codes are generally used for this purpose. ECC memories with SEC-DED capabilities are increasing in popularity, especially for mission-critical applications running in dependable systems. Many recent PC-servers, e.g., the IBM Blade server, the Dell PowerEdge server, etc., have a ChipKill memory architecture [Dell 97], which supports ECC capabilities both in main memory and cache. There are many ways to construct SEC-DED codes and implement the corresponding ECC circuitry. Whereas earlier work on designing optimal SEC-DED codes focused on reducing delay and area, the main idea in this work is to use the degrees of freedom in selecting the parity check matrix to minimize power with little or no impact on area and delay. The power minimization method is applied to two popular SEC-DED codes: standard Hamming codes [Hamming 50] and odd-column-weight Hsiao codes [Hsiao 70]. Experiments on benchmark memory traces indicate that our proposed technique results in power reduction in memory ECC circuits with little impact on area and delay.

Chapter 3 discusses a method for designing low power parity check codes for concurrent error detection (CED). CED involves detecting errors at the output of

a circuit while it operates. As technology continues to scale with smaller features sizes, lower power supply voltages, and higher operating frequencies, the soft error rate in logic circuits is rapidly increasing. CED provides a means to detect soft errors quickly before they have a chance to propagate and compromise the data integrity of a system, and is widely used in many applications to improve reliability. This chapter proposes an automated design procedure for synthesizing circuits with low power CED with parity check codes. The proposed idea is based on pre-synthesis selection of a parity-check code followed by structure constrained logic optimization to produce a circuit in which every single point faults is guaranteed to be detected. The main contributions of this technique are: (1) the use of a *k*-way partitioning algorithm combined with local search to select an optimal parity-check code, and (2) a methodology for minimizing power consumption in the CED circuitry. Experimental results indicate significant reductions in area overhead due to the new code selection procedure as well as the ability to find low power implementations for use in power-conscious applications.

Chapter 4 considers a technique of minimizing power dissipation in a scan-based circuit by re-ordering scan cells, which causes a relatively small increase of the circuit area. Scan-based testing is a technique of offline testing where the sequential elements of the circuit, which act as flip-flops in the normal operation of the circuit, are chained together in the test mode to create a scan chain. The test vectors to be applied are scanned into this scan chain, and it is also used to capture the output response from the circuit and scan it out. Scan-based DFT is popular due to good controllability and observability with low impact on performance, small area overhead and higher fault coverage. However, the shift operations during scan-in of test vectors and scan-out of test responses cause a lot of power dissipation in the combinational circuit connected to the scan chain. So, power reduction is very critical in scan-based designs. The suggested framework

provides the VLSI designer with a single design parameter that can be controlled to trade-off between the power reduced and the area overhead caused by the scan-chain re-ordering.

In Chapter 5, a technique for weighted pseudo-random test pattern generation is proposed that significantly reduces the power consumption in a circuit during testing. Linear feedback shift registers (LFSRs) are commonly used to generate pseudo-random test vectors that are applied to the circuit under test (CUT) to perform offline testing of VLSI circuits. One problem with this scheme is that typically a large number of pseudo-random test vectors are needed to reach an acceptable fault coverage, which results in a long test application time. To overcome this problem, several techniques for generating biased or weighted random patterns using LFSRs have been proposed, which achieve good coverage with a smaller number of random patterns. The main idea of the proposed approach is to identify which scan cells values remain fixed during application of a particular set of test vectors, and to fix the output values of those scan cells to eliminate transitions in the combinational circuit connected to those cells, thereby reducing power. We focus on a weighted pseudo-random test pattern generation scheme proposed by Pomeranz, *et al*., [Pomeranz 93] and propose an improvement to this algorithm that significantly reduces the power consumption in the CUT. To implement this algorithm in hardware, a new scan cell design, capable of performing bit-fixing, is proposed. It has all the capabilities of a normal scan cell and can additionally perform fixed-bit scan and capture, at the cost of a small increase in the area of the circuit.

Chapter 6 summarizes the work in this dissertation and discusses possible topics for future research.

Chapter 2

Reducing Power Consumption in Memory ECC Checkers

In this chapter, a method is proposed for reducing power consumption in memory ECC checker circuitry that provides SEC-DED. The degrees of freedom in selecting the parity check matrix are used to minimize power with little or no impact on area and delay. The power minimization method is applied to two popular SEC-DED codes: standard Hamming codes and odd-column-weight Hsiao codes. Experiments on actual memory traces of Spec and MediaBench benchmarks indicate that considering power in addition to area and delay when selecting the parity check matrix can result in power reductions of up to 27% for Hsiao codes and up to 41% for Hamming codes [Ghosh 04b].

2.1 Introduction

Error correcting codes (ECCs) are commonly used in memories to protect against soft errors and thereby enhance system reliability and data integrity [Chen 84], [Gray 00]. S*ingle-error-correcting* and *double-error-detecting* (SEC-DED) codes are generally used for this purpose. These codes are able to correct single-bit errors and detect double-bit errors in a codeword. There are many ways to construct SEC-DED codes and implement the corresponding ECC circuitry. While previous research has focused on minimizing area and delay in ECC circuitry, this chapter looks at minimizing power in addition to minimizing area and delay. By considering power during the design of ECC circuitry, significant reductions can be achieved at little or no cost in terms of area and delay.

As power has become a first-order design consideration, researchers have begun looking at techniques to reduce power consumption in error detection circuitry. While conventional low power design methodologies that have been developed for general circuits can be applied to the design of error detection circuitry in a straightforward manner, there are some special properties of error detection circuitry that can be exploited to further reduce power consumption. One such property is the fact that error detection circuitry typically contains large amounts of symmetry. For example, parity trees and two-rail checker trees are totally symmetric with respect to their inputs and thus allow complete freedom in the ordering of the inputs. The inputs can be ordered in any way with no change in the function of the circuit and no real impact on the area or delay. This property was first exploited to minimize power in [Favalli 97]. Favalli and Metra considered signal probability on a level-by-level basis to order the inputs in two-rail checkers to minimize power (the method can also be used for parity trees). In [Mohanram 02], spatial correlation among signals was used for input ordering in parity trees and Berger code checkers. A nice feature of both of these methods is that power is reduced essentially for free as there is no impact in terms of area or delay. The only cost is the time for computing the input ordering.

In [Rossi 02, 03], the problem of reducing power consumption for fault-tolerant buses with SEC codes was studied. The bus model that was used considers mutual capacitance effects and assumes transitions between all pairs of vectors are equally likely. The properties of both Hamming codes and dual rail codes with respect to power consumption were analyzed. Results in [Rossi 03] indicate that for small bus word sizes dual rail codes require less power, while for larger word sizes Hamming codes are better.

In this chapter, the focus is on reducing power consumption in memory ECC circuitry that provides SEC-DED. Such circuits are widely used in industry in all types of memories including caches and embedded memories. The key design

issue is selecting the code that is used. A *(n,k)* linear SEC-DED block code has *n* bits in each codeword consisting of *k* data bits and *n-k* check bits. The code can be represented by a parity-check matrix, ***H***, having *n-k* rows, one for each check bit, and *n* columns, one for each bit in the codeword. In order for the code to be SEC-DED, the ***H***-matrix must be formed in a way that the minimum distance between any codewords is 4. Two well-known methods for constructing a SEC-DED ***H***-matrix were described by Hamming [Hamming 50] and Hsiao [Hsiao 70]. Different ***H***-matrices result in different area, delay, and power. This chapter presents a method for selecting an ***H***-matrix that simultaneously minimizes power, area and delay. Once the ***H***-matrix has been selected, the corresponding ECC circuitry for implementing the code can be synthesized.

Another related work is [Kleihorst 01], where Hamming codes were designed with the goal of area minimization of the ECC checker in mind. In this work, we aim to minimize a joint function of area, delay and power while designing Hamming and Hsiao codes.

The chapter is organized as follows: Sec. 2.2 gives an overview of the proposed method; Sec. 2.3 discusses the ECC memory hardware details; Sec. 2.4 gives the details of the optimization algorithms used; Sec. 2.5 explains the experimental methodology, the results of which are discussed in Sec. 2.6; finally, Sec. 2.7 concludes our discussion and outlines promising areas of future work.

2.2 Overview of Proposed Method

The key idea of our approach is to select the ***H***-matrix in a way that minimizes power, area and delay in the ECC checker. The space of ***H***-matrices that provide SEC-DED capability is large. In [Hsiao 70], Hsiao showed that an ***H***-matrix that satisfies the following three constraints provides SEC-DED capability:

1. There are no all-0 columns.
2. Every column is distinct.
3. Every column contains an odd number of 1's (i.e., has odd *weight*).

Hsiao showed that by using minimum odd weight columns, the number of 1's in the ***H***-matrix could be minimized (and made less than a Hamming SEC-DED code). This translates to less hardware area in the corresponding ECC circuitry. Furthermore, by selecting the odd weight columns in a way that balances the number of 1's in each row of the ***H***-matrix, the delay of the checker can be minimized (as the delay is constrained by the maximum weight row).

In this work, power consumption is considered as an additional factor in selecting the ***H***-matrix. For odd-weight-column codes, there are two degrees of freedom in selecting the ***H***-matrix that can be used to reduce power with little to no impact on area and delay. The first degree of freedom is simply permuting the columns. This has no impact on area or delay as it does not change either the total number of 1's in the ***H***-matrix or the balancing of 1's among the rows. The second degree of freedom is in selecting the odd-weight-columns that are included in the matrix. To minimize area and delay, the smallest odd weight columns should be used first (i.e., weight-1, then weight-3, then weight-5, etc.). However, note that in general, only a subset of the largest odd weight columns will be used. For example, for a (72,64) odd-weight-column code, all $C_1^8 = 8$ of the weight-1 and all $C_3^8 = 56$ of the weight-3 columns will be used, but only 8 of the $C_5^8 = 56$ possible weight-5 columns will be used. Selecting which 8 of the 56 possible weight-5 columns are used in the ***H***-matrix is a degree of freedom that can be used for minimizing power with little to no impact on area or delay.

The amount of power that can be reduced using the degrees of freedom in selecting the ***H***-matrix will depend on the characteristics of the data stored in the

memory. The more correlated the data in successive memory reads and writes is, the more power can be reduced through careful selection of the ***H***-matrix. The switching activity (and hence power consumption) in the encoding and decoding logic corresponding to a particular ***H***-matrix depends on which bit transitions occur in the data between successive memory reads and writes.

Let us consider a very simple example to illustrate this point. Typically the high order data bit is more likely to be a 0 than a 1, whereas the low order data bit is more likely to have an even distribution between 0 and 1. Sparsity in higher order bits is a very common phenomenon for multimedia applications. In fact, special purpose compilers and architectures with support for variable bit width have been studied in order to exploit this characteristic of the multimedia applications [Stephenson 00]. Thus, since the low order bit is more likely to transition in successive memory accesses than the high order bit, it would be better that the low order bit correspond to a lower weight column in the ***H***-matrix and the high order bit correspond to a higher weight column in the ***H***-matrix. This would reduce the switching activity that occurs in the encoding/decoding logic and thereby reduce power. This is a simplistic example to show how selection of the ***H***-matrix can be used to exploit correlations in the data stored in the memory. More elaborate forms of spatial and temporal correlations in the data can be exploited with the proposed methodology.

How much correlation exists in the data stored in a memory will depend on the purpose and function of the memory. Some embedded memories for certain applications may have very correlated data and thus the proposed method for selecting the ***H***-matrix can be very effective in reducing power. Others may have less correlation. The types of data that are stored in different memories ranges broadly. Instruction caches and other memories that primarily contain instructions will tend to have a lot of correlation, as the frequency of execution of different instructions tends to be very skewed. Memories that contain a lot of numerical

data will tend to have a lot of spatial correlations among higher order bit positions as the range of the numerical values may be limited and/or skewed. Some embedded controllers and sensors may spend a lot of time executing in a loop and thus have a lot of temporal correlations. No matter what the nature of the memory is, not all transitions will be equally likely, so there will be some scope for power reduction using the proposed method. However, the actual amount of power reduction will depend on the extent of the correlation.

The proposed method consists of two steps. The first is to acquire information about the spatial and temporal correlations of the data in memory accesses. The second step is to use that information to select the ***H***-matrix for the odd-weight-column SEC-DED code. Information about the spatial and temporal correlations is acquired by analyzing a sample trace of the memory accesses for a typical workload. The application that will use the memory or a representative sample of the applications, if there are multiple applications, is simulated and a sample trace of memory accesses is obtained. The size of the sample should be chosen so that it is sufficiently representative of the typical workload. The spatial and temporal correlations among the data from the sample traces are then extracted so that the ***H***-matrix can be optimized for the typical workload of the memory. The resulting design of the ECC circuitry will then minimize the average power across the typical workload. For portable electronics this will help extend battery life.

Once the correlation information has been extracted, the second step of the proposed method involves selecting the ***H***-matrix. This problem is a non-linear optimization problem. In this work, two optimization techniques are investigated: simulated annealing (SA) and genetic algorithms (GA). Both techniques are described in Sec. 2.4. Experimental results showed that genetic algorithms outperformed simulated annealing for this problem. Genetic algorithms appear to be better suited to this problem as is discussed in Sec. 2.6.

2.3 ECC Memory Checkers

The goal of this work is to reduce the switching activity in the part of the ECC circuitry that is used most frequently, namely the parity generator block which is used on every memory access (both read and write). Fig. 2.1 illustrates the block-level design of a generic SEC-DED encoder/decoder for ECC memory. The left side of the figure is the processor interface where the relevant signals are *u_data*[63:0], representing the 64 bits of the processor data bus; *rw_n*, representing the memory read/write control signal; and *error-out*[1:0], the 2-bit error flag signal that is required to signal one of possible four error states: (1) no error, (2) correctable data error, (3) correctable parity error, and (4) detectable double error. The right side of the figure is the memory interface consisting of the 72-bit memory data bus *mem_data*[71:0]. The "Generate Parity Bits" block generates the parity bits to store with the processor data during a write cycle. During a read cycle, this block also generates the parity bits for the 64 data bits stored in memory. These generated parity bits are then compared with the stored parity bits to generate the syndrome. In this chapter, the focus is on selecting an SEC-DED code that minimizes power consumption in the parity generator block since that is the part of the circuit most heavily used.

Hamming codes and Hsiao codes are commonly used in ECC circuitry. The proposed optimization method is applicable to both of these kinds of SEC-DED codes. Note that the proposed method can be used for any memory word size.

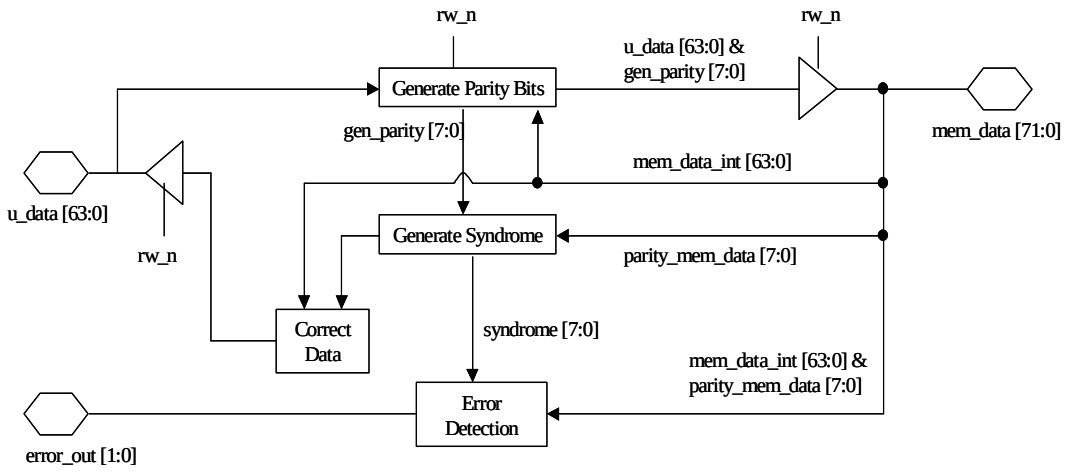

Figure 2.1: SEC-DEC Block Diagram (modified from [Xilinx 03])

2.4 Optimization Algorithms

In this chapter, two optimization algorithms that are known to give good performance for highly non-linear optimization problems, such as the one here, are investigated. One is simulated annealing (SA), and the other is genetic algorithms (GA). In this section, we give a brief description of both these techniques and how they were adapted to this domain.

2.4.1 Simulated Annealing (SA)

For Hamming codes, we consider the ***H***-matrix in the standard form and the only thing that can be varied to reduce power is the input ordering. For 64 inputs, there are 64! possible input permutations, which makes an exhaustive search of the input ordering with the lowest power dissipation intractable.

For Hsiao codes, along with input ordering, we have the additional flexibility of designing the ***H***-matrix. As explained in Sec. 2.2, selecting which 8 of the 56 possible weight-5 columns are used in the ***H***-matrix for a (72,64) code is a degree of freedom that can be used for minimizing the dissipated power. So, the search space is even larger in this case, having $(64! * C_8^{56})$ possible solutions.

To solve this large non-linear optimization problem, we applied simulated annealing [Kirkpatrick 83] to find a (local) optimum of the cost function, the details of which are described below.

2.4.1.1 Cost function

The cost function is modeled as a combination of the delay in the circuit, the size of the circuit, and the power dissipation in the circuit. It is a weighted linear combination of the following 3 components, which represent the different design objectives mentioned in Sec 2.2:

1. *Power dissipation*: The power dissipated during ECC checking, which is found by doing power simulation of the permuted inputs through the parity checker circuit. The *power goal* is minimization of this dissipated power.
2. *Size of circuit*: The number of total gates in the circuit, obtained by performing multiple-output logic minimization of the ***H***-matrix equations, using 2-input XOR gates. The *circuit-size goal* is reduction of the number of XOR gates in the total parity checker circuit.
3. *Delay in circuit*: The balance of delay in the circuit, measured by the variance between the depths of the XOR circuits corresponding to different parity equations. A checker circuit with minimum delay would have a perfect balance of depth between the XOR networks implementing each parity output, and would

correspond to 0 variance in the depth of the XOR networks. So, the *timing goal* is minimizing the variance in delay between the XOR networks corresponding to the different parity bits.

2.4.2 Genetic Algorithm (GA)

Genetic algorithm (GA) is another popular non-linear optimization tool, useful for such large-scale non-linear optimization problems [Holland 75]. In GA, each possible solution to the problem is encoded as a gene. An initial population of P random genes is considered, from which a next generation of P genes is created by *crossover* and *mutation* operations. We considered a variant of GA where the top E best genes (*elites*) at each generation are directly copied into the next generation, thus preserving the best E solutions found so far: this GA principle is called *elitism*.

For the purpose of illustration, we will consider $n = 64$ in this section. Note that all these methods can be generalized to work for architectures of other sizes.

2.4.2.1 Overall GA algorithm

The algorithm in Fig. 2.2 outlines the overall GA algorithm that we use.

2.4.2.2 Gene representation

For Hamming codes, each solution corresponded to a particular input permutation. The *inputGene* corresponding to this is encoded as a string of the mapping for the input memory bits positions. For example, for $n = 64$, one possible permutation could be represented in the *inputGene* by the string

Input: Initial population of *K* random genes, the number of elites *E*, the number *mutant* children *M*, the number of *unfit* parents *U*, the number of generations *G*.

Output: gene with maximum fitness (minimum cost)

Algorithm:

1. i = 0.
2. The *K* parent genes are sorted in decreasing order of fitness (increasing order of their cost).
3. Top *E elite* parents are copied into the next generation.
4. *M* children are created by direct *mutation* of the *E elites.*
5. Bottom *U* parent genes are rejected as *unfit*
6. Remaining *K-E-M* children are created by *crossover* between any 2 parents that are not *elites* and not *unfit*.
7. i = i + 1
8. If (i < G), goto Step 1. Else, return elite with maximum fitness (minimum cost).

Figure 2.2: Outline of Genetic Algorithm

"2,3,1,4,5…63,64", representing the permutation where the 1st memory bit position is mapped to the 2nd input in the checker circuit, the 2nd bit is mapped to the 3rd input, the 3rd bit is mapped to the 1st input, and the other memory bits are mapped to their corresponding circuit inputs.

For Hsiao codes, each solution also contains an additional component that we call the *matrixGene*, representing the design of the ***H***-matrix. In our 64-bit architecture example, we first index the 56 possible weight-5 columns in increasing order of their binary representation. The *matrixGene* is represented by the indices of the 8 weight-5 columns out of the 56 possible ones that are selected to fill up the last 8 positions of the ***H***-matrix (after filling up the first 64 with all possible weight-1 and weight-3 columns). So, a possible *matrixGene* would be "1,4,6,9,11,34,53,55", representing the indices of the particular weight-5 columns

selected while creating the ***H***-matrix. In the general case, for architectures of other sizes, the *matrixGene* would have a representation of a similar design choice of selecting some columns from a total set of possible odd weight columns.

In the case of Hsiao codes, the GA algorithm performed simultaneous *crossovers* and *mutations* of both the *inputGene* and the *matrixGene*. The *fitness* of a composite gene, comprised of the *inputGene* and the *matrixGene*, was considered to be the inverse of the total cost calculated as shown in Sec. 2.4.1.1, so that genes with less cost ended up being "more fit".

2.4.2.3 Mutation operation

The *mutation* operation for the *inputGene* creates a child from a single parent, by choosing two input index mappings at random in the parent gene and swapping them. For example, swapping the 1st and 4th positions in the example gene considered above would generate the mutant *inputGene* "4,3,1,2,5…63,64" from the parent "2,3,1,4,5…63,64".

In the case of Hsiao codes, the *mutation* operation for the *matrixGene* creates a child from a single parent by selecting a column index at random and removing it from the selected set, bringing in a column from the unselected set. For example, swapping out the weight-5 column having index 3 and swapping in the column with index 4 in the example gene considered above would generate the mutant child "1,3,6,9,11,34,53,55" from the parent *matrixGene* "1,4,6,9,11,34,53,55".

2.4.2.4 Crossover operation

The *crossover* operation for the *inputGene* creates a child from two parents, trying to incorporate good features of both. We chose a *crossover* function where

the mean of the positions in the two *inputGenes* was first computed, and the child was created by considering the sorted indices of the computed mean. In our example, the *crossover* between "3,1,2,4,5…63,64" and "1,2,3,4,5…63,64" would generate an intermediate mean [2.0,1.5,2.5,4.0,5.0…63.0, 64.0], for which the corresponding sorted indices would be "2,1,3,4,5…63,64" (where the ties between same values are broken arbitrarily).

Notice that in this crossover function, the child has the common feature of both the parents, i.e., matrix bit positions 4-64 are mapped to circuit input 4-64. If both the parents had low cost and this was a feature responsible for it, then the child would also inherit this feature.

In the case of the *matrixGene*, the *crossover* function appends the *matrixGenes* of both the parents (representing the selected columns) with the indices of the unselected columns. Then, an average and index-sorting operation similar to the *inputGene* is performed, after which the first 8 positions of the result are selected to get the *matrixGene* of the child. It can be easily shown that this operation is a valid *crossover* function for the subset-selection problem that underlies the design of the ***H***-matrix for Hsiao codes.

2.5 Experimental Methodology

We ran experiments on 5 sample programs from the Spec 1995 and 2000 architecture benchmark suites: compress, perl, go, gcc and anagram, and 5 benchmarks from the MediaBench multimedia benchmark suite [Lee 97]: decode, encode, epic, cjpeg and rawcaudio. We used the architecture tool *SimpleScalar* [Burger 96] to simulate a 64-bit architecture, and for each program all the memory read and write accesses were recorded. These memory traces are

the inputs to the "Generate Pariy Bits" block of the ECC checker circuit, which generates 8 parity-check bits corresponding to each 64 bit-wide memory word.

During estimation of the cost of each solution in the SA and the GA algorithms, the circuit corresponding to each ***H***-matrix was synthesized by multiple-output logic minimization with 2-input XOR gates as basic components using *sis* [Sentovich 92].

For each benchmark, the best input permutation and ***H***-matrix was obtained for both the Hsiao code and the Hamming code. The SA algorithm was initialized from a random solution, and the temperature was increased until the system "melted" [Szu 87]. Subsequently, the temperature was reduced in a Cauchy schedule and annealing was performed for 500 time-steps. The GA algorithm was run with the following parameters: size of population $K = 250$, number of elites $E = 5$, number of mutant children $M = 50$, number of filtered unfit parents = bottom 100, number of generations $G = 200$.

For both SA and GA, the performance of the final best solution of minimum cost was compared to 100 random solutions. For Hamming code, this corresponded to 100 random input orderings of the standard form Hamming code, whereas for the Hsiao code this corresponded to 100 random minimum odd-weight-column ***H***-matrices having a random input ordering. These random solutions emulate the convention design procedure that arbitrarily selects a code with minimum area and delay, but with no consideration of power.

2.6 Results

Tables 2.1 (a) and 2.1 (b) show the results of running the GA and SA algorithms on the 10 benchmarks. The first 5 benchmarks are Spec benchmarks, and the next 5 are from MediaBench. The combined cost function was used in all

these cases so that circuit size, delay and power were simultaneously minimized. We estimate power by the number of transitions in the outputs of the XOR gates of the checker circuit, the size of the circuit by the number of XOR gates in it, and the maximum delay in the circuit by the maximum level among the XOR networks implementing each parity equation. Note that it was assumed that the inputs to the checker are synchronized coming from a register and glitch power was not considered.

Hsiao and Hamming codes were studied as the two underlying SEC-DED codes of the ECC checker. For both GA and SA, we compare the number of transitions in the ECC checker circuit of the best solutions with the average (of 100) random number of transitions and the worst (out of 100) random number of transitions.

As can be seen from the results in Table 2.1 (b), GA gave 12% to 27% power reduction on the different benchmarks with respect to the average random transitions, and 14% to 34% reduction with respect to the worst-case random transitions. For this experiment, GA has much better performance than SA, which gave 1% to 14% power reduction with respect to average random, and 2% to 22% improvement compared to worst-case random. One possible reason for the better performance of GA over SA in this case could be that for Hsiao codes, the total power in the circuit is a highly non-linear and discontinuous function of the input ordering and choice of $\boldsymbol{H}$-matrix. Due to this, the gradient may not be well defined at every point on the cost function. So SA, which essentially performs a non-greedy gradient descent, does not perform very well. In comparison, GA's are known to perform well for cost functions with such characteristics. Moreover, the best E solutions found at every step of GA (E = 5 in our experiments) are deterministically remembered by using *elitism*, which is an added advantage of GA over SA in this case.

In general, the results show that power savings in GA increase with increasing size of the benchmark traces. A possible reason for this could be that the inputs get more correlated as the size of the benchmark traces increases, thereby giving more scope to the GA algorithm to perform better optimization.

Fig. 2.3 shows the characteristics of the 4 representative benchmarks from the set of 10 that we have considered, two each from Spec and MediaBench. The plots on the left show the signal probabilities in the columns of the memory trace matrix of the benchmark programs, sorted in increasing order. To generate each graph, the signal probability (i.e., probability of 1's) is computed for every column, and then the columns are sorted in increasing order of signal probability from left to right in the plot. For perfectly random inputs, each column in the input trace matrix would have 0.5 fraction of 1's, since 1's and 0's would be equally probable. The skewness of this distribution demonstrates that there is an uneven distribution of the number of 1's in different columns of the input memory trace matrix. The power optimization algorithms exploit this during re-ordering.

The plots on the right in Fig. 2.3 are histograms that show the pairwise correlations between the columns. The histograms were constructed as follows: for each pair of columns in the input memory trace, we counted how many transitions between 1 and 0 (and vice versa) would occur if we placed an XOR gate between the two columns. The histogram counts how many column pairs have their fraction of transitions in each bin range. If any two columns in the input trace matrix were independent, then the proportion of transitions would be 0.5 x 0.5 = 0.25. The corresponding histogram would have all the frequency concentrated at the 0.25 bin. In this case, the distribution of histogram frequencies in multiple bins for different benchmarks, ranging from 0.04 to 0.34, demonstrates that there is significant useful correlation between the input columns, which is useful for power optimization.

We also ran experiments on the (72,64) Hsiao code where we individually considered only the power, circuit size, and circuit delay components of the GA cost function. The results of these ablation experiments are shown in Tables 2.2 (a) and 2.2 (b). Note that in each individual optimization, we obtain values that are generally better than the corresponding values obtained using the overall cost. For example, for the benchmark program encode, the overall cost minimization gave power savings of 14%, a circuit size of 172 gates and the maximum number of levels to the output was 7. In comparison, individual minimizations of power, circuit size and maximum number of levels gave 15% power savings, 171 gates and 6 maximum number of levels respectively, which are individually better than their corresponding results for the combined cost function. However, when one component in the cost function is individually minimized, the other two components can have highly non-optimal values since the cost function does not consider them at all during the optimization process (as shown by the negative power savings, i.e., *increase* in power dissipated with respect to random, in many cases, if only delay is minimized). The overall cost function gives a good tradeoff between minimization of power and satisfaction of the other design requirements. Note that the weights of the 3 components of the cost function gives the designer the flexibility to incorporate specific design choices, e.g., more importance to power minimization over circuit size or delay minimization.

An interesting observation in the ablation study is that for compress and go, the power reduction for the combined cost function is better than the power reduction for individual power minimization. This apparently seems like an anomaly, but it can be explained as follows. Since power is a highly non-linear function of the input permutation and the choice of ***H***-matrix, there are many local minima in this function. The reduction of only the power component limits the search of the GA, and can cause the GA to sometimes get stuck in local minima. In these cases, using the combined cost function can help the

optimization algorithm to get out of such local minima and get to a better optimum, as we see in the case of the compress and go benchmarks. So, apart from finding a good overall minimum of the various design components (power, circuit size, maximum delay), using the combined objective function also facilitates the GA algorithm to get do a better exploration of the search space, which enables it to avoid bad local minima more effectively in some cases.

In the next set of experiments, we ran the GA algorithm on the (72,64) standard Hamming code, for each of the benchmark circuits. Table 2.3 shows the power savings on the Hamming code, which are between 5% and 41% for the different benchmark circuits, are better than the corresponding power savings for the Hsiao code in most cases. One possible explanation for that is that the Hsiao code is well optimized in terms of balance of gates between the different parity circuits. In comparison, Hamming codes have a large skew in the number of XOR gates in different parity equations, which can be exploited by the optimization algorithm more effectively. Moreover the standard Hamming code typically produced larger circuits when synthesized, which also gave the GA algorithm a larger search space where it could produce solutions significantly better than random.

2.7 Conclusions

Overall, our experiments demonstrate that there is significant correlation among memory traces for the benchmark applications we studied, and that optimizing the input permutation and the design of the ***H***-matrix of the memory ECC checker using GA with a combined cost function gives us significant power reduction, while simultaneously minimizing the overall size of the circuit and the circuit delay. Note that both SA and GA are relatively slow optimization

procedures, but the optimization is performed offline and it is the optimized ***H***-matrix that is deployed in the online error correction phase. So, speed of the optimization algorithm is not a major issue for the problem we are studying, implying that more sophisticated search or optimization techniques could be employed if necessary.

An area for future work is to extend the technique described here to handle memory ECC for the ChipKill server architecture [Dell 97] and for other error-correcting codes, e.g., Reed-Solomon codes, Fire codes, etc. For some of these codes, the proposed scheme will have be modified to handle certain characteristics of the codes, e.g., for byte error-correcting codes, *b*-byte column groups of the ***H***-matrix would have to be permuted instead of the columns being permuted directly. However, it would be relatively straightforward to frame the problem of finding the "best" ***H***-matrix in these cases as an optimization problem, which can be solved by simulated annealing (SA) or genetic algorithms (GA).

Another interesting area of future research is the study of how the presence of caches would affect the correlation in the data input to the ECC memory, and whether there is any systematic pattern there that can be exploited by the optimization algorithms.

Table 2.1 (a): Power results of SA on Hsiao code with overall cost function

Name	Memory trace size	Average random solution #transition	Worst random solution #transition	SA solution #transition	SA power reduction	
					w.r.t. average random	w.r.t. worst random
gcc	187089	6760149	7353330	6414782	5.1%	12.8%
go	118897	5449560	5852080	5156709	5.4%	11.9%
anagram	94041	4953000	5097104	4589887	7.3%	10.0%
compress	72193	1518005	1622810	1383552	8.9%	14.7%
perl	27657	1186947	1228706	1159770	2.3%	5.6%
epic	470633	17111347	18898620	14700707	14.1%	22.2%
cjpeg	18273	935956	975516	920797	1.6%	5.6%
encode	7569	399527	417434	394907	1.2%	2.3%
decode	4745	237972	251488	234402	1.5%	6.8%
rawcaudio	2233	141921	147616	121390	14.5%	17.8%

Table 2.1 (b): Power results of GA on Hsiao code with overall cost function

Name	Memory trace size	Average random solution #transition	Worst random solution #transition	GA solution #transition	GA power reduction	
					w.r.t. average random	w.r.t. worst random
gcc	187089	6760149	7353330	5042942	25.4%	31.4%
go	118897	5449560	5852080	4253540	22.0%	27.3%
anagram	94041	4953000	5097104	4358206	12.0%	14.5%
compress	72193	1518005	1622810	1222844	19.4%	24.7%
perl	27657	1186947	1228706	1014818	14.5%	17.4%
epic	470633	17111347	18898620	12445636	27.3%	34.2%
cjpeg	18273	935956	975516	759018	18.9%	22.2%
encode	7569	399527	417434	329548	17.5%	21.1%
decode	4745	237972	251488	195610	17.8%	22.2%
rawcaudio	2233	141921	147616	118396	15.6%	19.8%

Table 2.2 (a): Overall results of GA on Hsiao code with individual cost functions

Benchmark Name	Minimize power only			Minimize delay only			Minimize circuit size only		
	Power saved	Ckt. size	Max levels	Power saved	Ckt. size	Max levels	Power saved	Ckt. size	Max levels
gcc	27.5%	172	7	-4.0%	172	7	0.9%	171	8
go	19.8%	172	8	-1.9%	173	7	4.6%	171	8
anagram	14.1%	172	8	1.9%	172	7	1.7%	171	7
compress	18.6%	174	7	3.3%	174	7	1.0%	171	7
perl	14.6%	172	7	0.4%	173	7	2.7%	171	8
epic	30.5%	172	8	-1.2%	172	7	3.3%	171	7
cjpeg	18.1%	172	8	-1.4%	172	7	1.7%	171	8
encode	15.2%	173	7	-2.7%	173	6	4.2%	171	7
decode	18.1%	172	7	-1.4%	173	6	2.5%	171	8
rawcaudio	17.7%	173	7	1.4%	173	7	3.7%	170	7

Table 2.2 (b): Overall results of GA on Hsiao code with combined cost function

Benchmark Name	Minimize combined cost function		
	Power saved	Ckt. size	Max levels
gcc	25.4%	171	7
go	21.9%	172	7
anagram	12.0%	174	7
compress	19.4%	175	7
perl	14.6%	172	7
epic	27.3%	173	7
cjpeg	18.1%	172	8
encode	14.2%	172	7
decode	17.8%	171	7
rawcaudio	15.6%	172	7

Table 2.3: Results of GA on Hamming code with overall cost function

Benchmark Name	Average random solution #transition	GA solution #transition	Hamming code power reduction
gcc	5793700	3408358	41.2%
go	4663511	2764384	40.7%
anagram	4120663	2810202	31.8%
compress	1161115	1097838	5.4%
perl	983159	710168	27.8%
epic	14792749	8628410	41.7%
cjpeg	783432	543062	30.7%
encode	336843	223354	33.7%
decode	201138	127272	36.7%
rawcaudio	120058	79654	33.6%

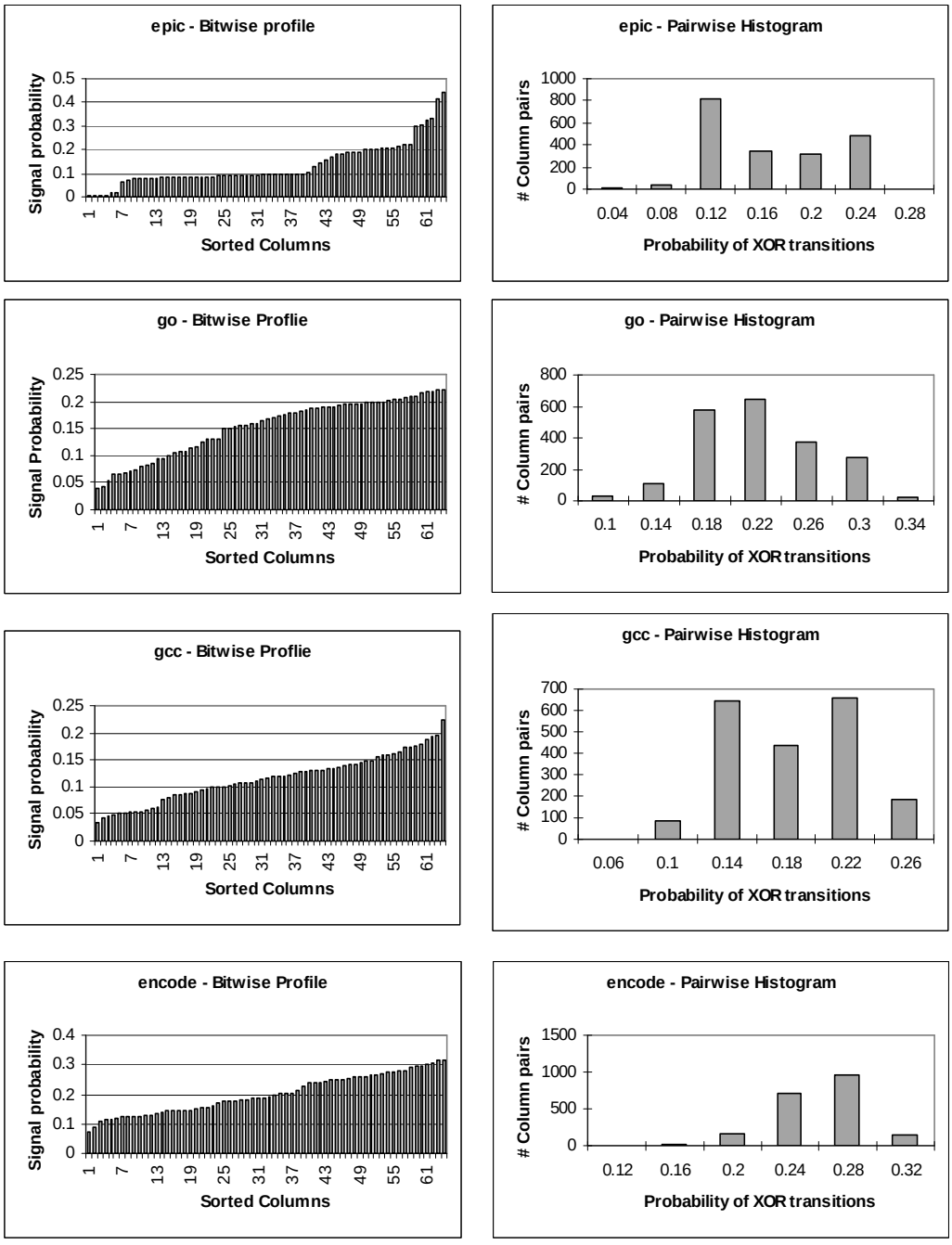

Figure 2.3: Characteristics of benchmark traces

Chapter 3

Synthesis of Low Power CED Circuits Based on Parity Codes

In this chapter, an automated design procedure is described for synthesizing circuits with low power concurrent error detection. It is based on pre-synthesis selection of a parity-check code followed by structure constrained logic optimization that produces a circuit in which all single point faults are guaranteed to be detected. Two new contributions over previous work include (1) the use of a k-way partitioning algorithm combined with local search to select a parity-check code, and (2) a methodology for minimizing power consumption in the CED circuitry. Results indicate significant reductions in area overhead due to the new code selection procedure as well as the ability to find low power implementations for use in power conscious applications [Ghosh 05].

3.1 Introduction

Concurrent error detection (CED) involves detecting errors at the output of a circuit while it operates. As technology continues to scale with smaller features sizes, lower power supply voltages, and higher operating frequencies, the soft error rate in logic circuits is rapidly increasing [Shivakumar 02]. CED provides a means to detect soft errors quickly before they have a chance to propagate and compromise the data integrity of a system. CED is widely used in many applications to improve reliability.

One way to implement CED is to encode the outputs of a circuit with an error detecting code and have a checker that monitors the outputs and gives an error

indication if a non-codeword occurs (as illustrated in Fig. 3.1). The check-bit generator is the parity prediction circuit that calculates the parity bits directly from the circuit inputs. The parity checker is self-checking so that any error that occurs in the checker itself is detected. One commonly used error detecting code is a parity-check code. A parity-check code is a linear code in which each parity check bit checks the parity over a group of output bits. Two special cases of a parity-check code are single-bit parity where there is a single parity bit checking all the outputs of the functional logic and duplication where there is a parity bit checking each output of the functional logic.

A number of techniques have been proposed for automated design of circuits with CED based on parity-check codes. There are two basic approaches. One is to first synthesize the functional logic and then select the parity-check code (post-synthesis code selection). The other is to select the parity-check code and then synthesize the functional logic with structural constraints to ensure high coverage (pre-synthesis code selection). For post-synthesis methods, the goal is to select a parity-check code that provides high coverage while minimizing the complexity of the parity prediction logic (i.e., check bit generator). Since the functional logic circuit is known up front, the code can be selected so that it detects all the output error combinations that can arise due to a specified fault class [Sogomonjan 93], [Goessel 93]. Recent work in [Almukhaizim 04] has investigated the use of fast entropy estimation techniques to find parity-check codes with less complex parity prediction logic. If 100% coverage is not necessary, then the complexity of the parity prediction logic can be reduced. This can be accomplished by using a self-dual complement [Saposhnikov 96] or disabling the parity check for some input combinations [Mohanram 03].

For pre-synthesis methods, the selection of the parity-check code places constraints on the structure of the functional logic. The goal is to find a parity-check code that minimizes the overall area considering the functional logic,

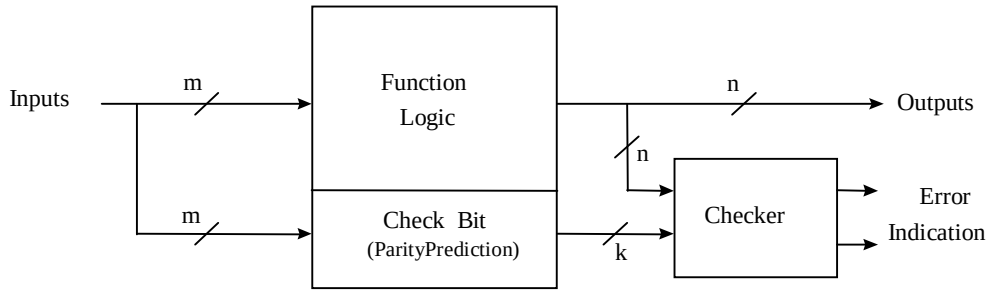

Figure 3.1: Concurrent error detection using parity-check code

check-bit generator, and parity checker. Techniques in [De 94] and [Touba 97] have been proposed to constrain the structure of the functional logic so that a simpler parity-check code can be utilized to provide 100% coverage of single-point faults. By trading off structural constraints on the functional logic (which may result in it being larger) to get a simpler parity-check code (which reduces the size of the parity prediction logic), the overall size of the circuit with CED can be reduced. The key issue here is how to best select the parity-check code that optimizes this tradeoff. The approach in [Touba 97] uses a simply greedy algorithm that starts with the duplication code and iteratively merges parity groups as long as the merging causes the overall area to decrease. The number of parity groups is determined automatically in this greedy algorithm by the convergence criterion of merging until no more area reduction is possible. The drawback of this approach is that it can easily get caught in local minima since no look-ahead information is used during each merge.

While previous work in automated design of circuits with CED based on parity-check codes has focused on minimizing area, this work investigates minimizing power dissipation. In many low power applications, including hand-held devices, mobile computing, laptop computers, etc., minimizing power is a

first-order design issue. This chapter presents a new approach for selecting a parity-check code that provides the best tradeoff between structural constraints on the functional logic and complexity of the parity prediction logic to reduce the overall power of the circuit with CED. The problem is reformulated as a *k*-way partitioning problem that overcomes the limitations of earlier approaches to identify more optimal parity-check codes. Experiments on benchmark circuits demonstrate that the proposed approach is able to reduce the power dissipation of the CED circuit as well as its area compared to previous techniques.

The chapter is organized as follows: Sec. 3.2 gives an overview of the overall scheme, Sec. 3.3 gives details of the proposed 2-phase algorithm, Sec. 3.4 outlines the results of experiments on benchmark circuits, and finally Sec. 3.5 concludes the chapter.

3.2. Overview of Proposed Technique

In the proposed technique, a parity-check code is used to detect all single point faults in the circuit. To ensure this, one has to be careful about logic sharing while synthesizing the logic circuit so that single point faults in the circuit cannot cause errors that get masked and go undetected. We use the structure constrained logic synthesis algorithm of [Touba 97] for synthesizing the logic circuit. The basic idea of this method is that two outputs assigned to the same parity group should not share any logic, because that may allow a single error in the shared logic block to be propagated to both the outputs, resulting in a two-bit error that would not be detected by the parity code. The structure constrained logic optimization algorithm essentially enforces constraints on logic sharing that are necessary to ensure that no undetectable errors are caused by single point faults.

The power consumption in each of the CED circuit components depends on the type of parity check code used for concurrent error detection. If the duplication code is used (i.e., each output is put in its own parity group), then there are *n* parity bits for *n* logic outputs. In that case, the parity prediction circuit is a duplicate of the original logic circuit and can have significant power overhead. However, since there are no logic-sharing constraints between outputs in each parity group, the structure constrained logic optimization can share a lot of logic to reduce the power dissipation in the function logic. On the other hand, if all the outputs were put into one parity group, then the parity prediction logic would in general be simpler and have less power dissipation. But then the structure constrained logic optimization algorithm would not be able to share a lot of logic when synthesizing the function logic, which could result in increased power dissipation in the function logic. Thus, we see that there is a tradeoff between the power dissipation of the parity prediction circuit and the functional logic circuit, for which the proper choice of the parity check code is essential to minimize the overall power of the circuit with CED.

The problem of finding an optimal parity check code is equivalent to finding the optimal grouping of the outputs of the function logic such that the power of the circuit with CED is minimized. This suggests a formulation of the problem as a *k*-way partitioning of the logic outputs into *k* groups so that the power reduction from merging the outputs in each group is maximized.

3.3. Proposed Algorithm

In this work, we propose a 2-phase non-greedy algorithm of *k*-partitioning and local search. This avoids the problem of a greedy algorithm getting stuck in bad

local optima. The algorithm uses a power-based cost function that estimates the power reduction due to merging two parity groups at a time.

3.3.1. Power-based Cost Function

Merging two parity groups affects the power dissipation in each component of the CED circuit. The reduction in power of the checker and parity prediction circuit can be calculated by the difference in the power dissipated before and after the merging. However, merging of the parity groups also reduces the amount of logic sharing in the functional logic during structure-constrained logic synthesis, which results in more power dissipation. The overall power cost function is the difference of these two components:

> *Effective Power Reduction = (Power reduction in checker circuit and parity prediction circuit) – (Power increase from decreased logic-sharing in logic circuit)*

During logic synthesis, the circuit is represented by a Boolean network [Sentovich 92] in which each node represents a two-level logic function and an edge exists from node *A* to node *B* if node *A* is a fanin to node *B*. To estimate the effective power reduction, we consider the Boolean network of the parity prediction circuit for the parity code before merging. The power dissipated in each node of the Boolean network is estimated by decomposing the node into 2-input gates, assuming equiprobable input values to the primary inputs of the Boolean network, and calculating the resulting switching probability at each gate output weighted by its capacitive load. We merge the two parity groups under consideration by removing the two corresponding nodes from the Boolean

network for the parity prediction circuit and adding a new node obtained by taking an XOR of the logic functions of those nodes. The difference in load-weighted switching probabilities between the new node and the two original nodes gives us an estimate of the power reduction due to merging.

To estimate the second component of the power cost function, we calculate the total power dissipated in the function logic nodes that are reachable from the outputs of both the parity groups. This gives an estimate of the power increase due to decreased sharing in the logic circuit, since the nodes that are reachable from both the parity groups cannot be shared when the functional logic is synthesized with constraints. Since that logic cannot be shared, the total power consumption will increase by that amount.

The difference between the first and the second components gives us the power cost function, i.e., the effective power reduction due to merging two parity groups. For larger circuits, we can use sampling instead of exact power estimation for computational efficiency.

Note that the actual power dissipation in the parity prediction circuit depends on which cell-library is used to map the logic gates to cells. As an approximation, the proposed procedure decomposes the parity prediction circuit into 2-input gates when estimating power. We also did not consider changing the ordering of the operations in the parity tree to reduce the power consumed in this circuit component – schemes such as the one described in [Mohanram 02] could be used to reorder the parity operations and further reduce power dissipation.

3.3.2 *k*-partitioning and Local Search

The proposed algorithm partitions the logic outputs into parity groups corresponding to the lowest-power parity check code. It finds the best number of

parity groups *k* by searching all values of *k* between *1* and *n*, where *n* is the total number of logic outputs. For each value of *k*, a 2-phase technique is used for partitioning the logic outputs into parity groups. The details of the algorithm are given in Fig. 3.2.

In the first phase, a fast *k*-partitioning technique is used to create *k* good initial groups. Given a particular value of *k*, the algorithm chooses *k* outputs that have the minimum power reduction due to mutual pairwise merging and initializes the *k* partitions with these outputs. These initial outputs are found using the *farthest-first* algorithm [Hochbaum 85], where the basic idea is to get *k* points out of *n* that are mutually "far" from each other. In farthest first traversal, an initial point is chosen at random. The next point is selected to be farthest from it using a particular distance measure and is added to the traversed set. The remaining points are selected to have maximum distance from the traversed set, where we use the standard notion of distance of a point *x* from a set *S*: $d(x,S) = \min_{z \text{ in } S} d(x,z)$. In this case, the distance between two logic outputs *a* and *b* was set to the inverse power reduction *1/power_reduction(a,b)* in merging the outputs, and farthest first traversal was performed on the logic outputs using this distance measure. So, when we find the *k* initial outputs by farthest first traversal, they are put in different partitions since there will not be a substantial power reduction by merging any two of these outputs. The other outputs are then sequentially merged with their "nearest" partition using a *nearest-neighbor* assignment scheme, that effectively gives the maximum power reduction for each assignment of an output to a group.

In the second phase, using a local-search refinement technique further refines these initial partitions. Local search is a method of perturbing a solution so as to help it go out of potential local minima. We use a variant of local search where a series of local refinements are performed by considering each output in turn. The local refinement considers removing each output from its current partition and

placing it in a new partition, and the resulting power reduction is calculated. For a given output, it finds the movement that gives the maximum power reduction, and the output is removed from the old partition and merged with the new partition. Every output is considered in turn for this local refinement step, and the process is continued iteratively until no more movements give further power reduction. In the end, the outputs in every partition form a parity group.

Since both these phases are computationally efficient, they can be applied to do a non-greedy search over all possible values of *k* from *1* to *n*, thereby exploring the exponential-size space of all possible parity groupings more effectively. The value of *k* that gives the best overall reduction in power and area is used as the number of parity groups, and the corresponding *k*-partitioning is used to synthesize the parity prediction and function logic circuits.

3.4 Experimental Results

For our experiments, the proposed algorithm was run on combinational circuits from the *MCNC* benchmark suite. The *k*-way partitioning, local search, and structure-constrained logic optimization algorithms were implemented in *SIS-1.2* [Sentovich 92]. The internal BDD power simulator in *SIS-1.2* was used to do power simulation using a zero-delay model. Power was measured by the weighted switching probabilities of all the nodes of the circuit with CED, where the weight for the switching probability of a node corresponds to the capacitive load of that node. We used two area estimates – the total number of factored form literal counts and the cell area of the circuit with CED after structure-constrained logic synthesis. The mapping was performed using 2-input cells from the *mcnc.genlib* library and the cell area numbers are given in units of $1000\lambda^2$, where λ is the minimum size in the technology.

Input: Logic circuit with *n* outputs.
Output: Number of parity groups and parity grouping of the logic outputs corresponding to the optimal-power parity check code.

Algorithm:
1. Initialize
   best_power = 0
   best_num_groups = 0
   best_grouping = NIL
2. For *k = 1* to *n*
3. Initialize *k* partitions using *farthest_first_init(k)*
4. Assign outputs to the initial partitions using *nearest_neighbor_assign(k)*
5. Refine the partitioning obtained using *local_search_refine(k)*, store resulting partitioning in *current_grouping*
6. If total power reduction *power_reduced* in steps 3 and 4 is more than *best_power*, update
   best_power = power
   best_num_groups = k
   best_grouping = current_grouping
7. Return *best_num_groups* and *best_grouping*

Subroutines:

farthest_first_init(k)
1. Initialize *k* partitions with *k* logic output chosen using the *farthest-first* heuristic, using 1/*power_reduced(a,b)* as measure of distance between *a* and *b*

nearest_neighbor_assign(k)
1. for *i = 1* to *n*
2. If output *i* is not already assigned to one of the *k* initial partitions, then assign output *i* to the partition that is nearest to it, i.e., has the maximum power reduction on merging

local_search_refine(k)
1. for *i = 1* to *n*
2. Initialialize *best_reduction = 0*
3. for *j = 1* to *k*
4. *current_reduction* = power reduced by moving output *i* from its current partition to the new partition *j*
5. If *current_reduction > best_reduction*
   best_reduction = current_reduction
   best_new_partition = j
6. Move output *i* to partition *best_new_partition*
7. Return if *best_reduction = 0* for all outputs, else repeat steps 1-6

Figure 3.2: 2-phase algorithm for parity code selection

In the first experiment, we used the 2-phase algorithm for parity code selection with the proposed cost function that considers power reduction. Table 3.1 shows that our proposed 2-phase scheme of *k*-partitioning and local search refinement reduces power by as much as 34% on the benchmark circuits. As seen in Table 3.2, running the 2-phase algorithm with the power cost function also reduces area for all the benchmark circuits by as much as 35% as both area and power are correlated to some degree. The results in Table 3.1 also show that for some of the benchmark circuits the number of parity groups selected by our algorithm is different from that chosen by the greedy scheme in [Touba 97], demonstrating that the 2-phase algorithm is able to reach a better parity check code by a more effective search of the space of all possible parity groups.

We performed another experiment to explore in detail the effectiveness of each phase in the proposed 2-phase algorithm. As shown by the results in Table 3.3, the first phase (i.e., *k*-way partitioning only) alone gave some reductions in power, but using the 2-phase algorithm (i.e., *k*-way partitioning and local search) increased the amount of power reduction. This demonstrates that each of the 2 phases in the algorithm plays a significant role in selecting a low-power parity code.

Table 3.1: Comparison of power reduction of proposed 2-phase algorithm with greedy algorithm in [Touba 97]

Circuit Information			Greedy Algorithm [Touba 97]		Proposed Algorithm		Comparison
Name	PIs	POs	Number of Parity Groups	Power Dissipated	Number of Parity Groups	Power Dissipated	Reduction in Power (%)
misex1	8	7	3	463	3	305	34.1
wim	4	7	2	211	2	159	24.6
rd53	5	3	1	247	1	171	30.6
squar5	5	8	3	513	1	360	29.8
dc1	4	7	3	396	2	285	28.0
adr2	4	3	2	193	2	140	27.7
b12	15	9	4	589	2	521	11.6
rd73	7	3	1	716	1	518	27.7
misex2	25	18	3	624	4	516	17.4
bw	5	28	9	1375	2	1086	20.9
alu2	10	6	5	1527	4	1482	3.0
inc	7	9	2	740	2	573	22.5
5xp1	7	10	3	921	1	827	10.2

Table 3.2: Comparison of area reduction of proposed 2-phase algorithm with greedy algorithm in [Touba 97]

Circuit Information			Greedy Algorithm [Touba 97]		Proposed Algorithm		Comparison	
Name	PIs	POs	Number of Literals	Cell Area	Number of Literals	Cell Area	Reduction in Literals (%)	Reduction in Cell Area (%)
misex1	8	7	138	156	91	105	30.1	32.7
wim	4	7	69	82	50	57	23.2	30.5
rd53	5	3	55	73	35	47	27.4	35.6
squar5	5	8	139	174	93	120	26.4	31.0
dc1	4	7	101	117	65	78	30.8	33.3
adr2	4	3	48	56	30	37	32.1	33.9
b12	15	9	174	209	153	159	10.0	23.9
rd73	7	3	158	196	121	135	18.9	31.1
misex2	25	18	296	355	255	319	11.5	10.1
bw	5	28	442	548	362	426	14.6	22.3
alu2	10	6	417	512	409	469	1.6	8.4
inc	7	9	215	253	169	197	18.2	22.1
5xp1	7	10	247	289	211	240	12.5	17.0

Table 3.3: Breakdown on power reduction for each phase of the proposed 2-phase algorithm

Circuit Information			Greedy Algorithm [Touba 97]	After phase 1	After phase 2	Comparison	
Name	PIs	POs	Power Dissipated	Power Dissipated	Power Dissipated	Power reduction after phase 1 (%)	Power reduction after phase 2 (%)
misex1	8	7	463	326	305	29.6	34.1
wim	4	7	211	166	159	21.3	24.6
rd53	5	3	247	187	171	24.2	30.6
squar5	5	8	513	399	360	22.1	29.8
dc1	4	7	396	298	285	24.9	28.0
adr2	4	3	193	140	140	27.7	27.7
b12	15	9	589	551	521	6.5	11.6
rd73	7	3	716	564	518	21.2	27.7
misex2	25	18	624	546	516	12.5	17.4
bw	5	28	1375	1108	1086	19.4	20.9
alu2	10	6	1527	1489	1482	2.4	3.0
inc	7	9	740	577	573	22.1	22.5
5xp1	7	10	921	882	827	4.3	10.2

3.5 Conclusions

This chapter presents a new 2-phase algorithm for synthesis of low-power concurrent error-detecting circuits based on parity codes, which outperforms a previously known parity-code selection technique. Along with reducing the power of benchmark circuits with CED by as much as 34%, the 2-phase algorithm is also able to simultaneously reduce their area by as much as 35%.

Chapters 2 and 3 have described methods of power reduction in two applications of online testing for concurrent error detection. The next two chapters will discuss techniques to reduce power dissipation in offline testing.

Chapter 4

Joint Minimization of Power and Area in Scan Testing by Scan Cell Reordering

This chapter describes a technique for minimizing power dissipation in scan testing, a popular method of offline circuit testing. The proposed method reduces power in scan-based circuits by re-ordering the scan cells in a scan chain, while also reducing the area overhead of the circuit compared to an arbitrary ordering of the scan cells. For a given set of test-vectors, we find the (locally) optimal re-ordering of the scan cells that minimizes a score function, which is a linear combination of the power and the area overhead. The score function has a trade-off parameter λ that can be used by the designer to specify the relative importance of area overhead minimization and power minimization. Our proposed greedy algorithm finds the best ordering for a given value of λ. The strength of our algorithm lies in the fact that we use a novel dynamic minimum transition fill (MT-fill) technique for the unspecified bits in the test vector. Experiments performed on the ISCAS-89 benchmark suite show a reduction in power (e.g., 70% for circuit s13207, λ = 500) as well as a reduction in layout area (e.g., 6.72% for circuit s13207, λ = 500) [Ghosh 03].

4.1 Overview

Two of the main drawbacks of scan testing [McCluskey 86] are the power dissipation and the area overhead. In CMOS circuits, power dissipation is proportional to the amount of switching activity that takes place [Devadas 95]. During scan testing, a much larger percentage of the scan cells will change value

in each clock cycle than during normal operations. This excessive switching activity during scan testing can cause power dissipation in the circuit to be very high. Another drawback of scan testing is the area overhead. As suggested in [Makar 98], one of the biggest components adding to the area overhead is the stitching wire between consecutive cells in the scan chains. Different techniques for controlling power dissipation during scan testing have been proposed in various research projects in the last decade, which include [Dabholkar 98], [Sankaralingam 00], [Girard 99A], [Gerstendörfer 99], [Chow 94], [Wang 97A], [Wang 97B], [Hertwig 98], [Wang 99], while various techniques for reducing the area overhead have been proposed in [Makar 98], [Lin 96].

In this chapter, we describe a technique for minimizing power dissipation during scan testing. Our technique is also capable of reducing the area overhead of the circuit, with respect to random ordering of the scan cells. For a given set of test-vectors, we find the (locally) optimal re-ordering of the scan cells that minimizes a score function, where the score function is a linear combination of the power and the area overhead. The score function has a trade-off parameter λ that can be used by the designer to specify the relative importance of area overhead minimization and power minimization – increasing λ causes a decrease in the power dissipation in the circuit, at the cost of increased area overhead. We propose a greedy algorithm for finding the best ordering for a given value of λ. The strength of our algorithm lies in the fact that we use a novel dynamic minimum transition fill (MT-fill) of the 'X' (i.e. unspecified) bits in the test vector. The method of doing "on-the-fly" MT-fill of the test vector matrix while calculating the optimal ordering gives us a better power reduction in the re-ordered matrix, details of which will be explained in Sec. 4.4.

We ran experiments on standard benchmark circuits and show power versus area overhead trade-off plots. These plots provide the designer with the flexibility

of giving more importance to minimizing power or area overhead, according to the design requirements, by choosing a suitable value of the design parameter λ. Our experiments show that with a proper choice of the parameter λ, our algorithm is quite effective in reducing the power in a circuit. For example, power in the circuit s13207 was reduced by 70%, for $\lambda = 500$. It is also capable of reducing the area overhead of the circuit, with respect to random ordering of the cells. For example, layout area overhead in the circuit s13207 was reduced by 6.72%, for $\lambda = 500$.

4.2 Background

In this section, we discuss some of the relevant background concepts used in the chapter.

4.2.1 Estimation of Power

In CMOS circuits, the predominant fraction of power is dissipated when circuit elements switch from logic 0 to 1 or vice versa. For a circuit-under-test (CUT), controlled entirely by the test vectors applied to it, the elements will switch value when the primary inputs change value or the scan cells change values. We assume that if the primary inputs of the CUT are directly controllable from the chip pins, then they are held constant during scan-in. Thus in this case, during scan-in, all switching activity is due to the transitions in the scan chain.

Let us consider a CUT with 4 scan cells and a vector 1001 being scanned in. Let the scan cells initially be 0000. At the first clock when the first input is scanned in, the state of the scan cells will be 1000. Thus the state of the first cell has changed from 0 to 1. This will cause other gates in the CUT to switch depending on the circuit. At the second clock when the next input is being

scanned in, the cells will be in a state 0100. Here both the first and the second cells have changed states. This continues until the complete test-vector has been scanned in. The test vector is then applied to the CUT and the output response is captured back in the scan chain. As the next scan vector is being scanned in, the transitions in the output response from the previous scan vector being scanned out will also cause switching activity. Thus we can divide the power dissipation during scan testing into:

- scan-in power - due to transitions in scan test vectors
- scan-out power - due to transitions in the output response being scanned out

The best way to estimate power during scan testing would be to do actual circuit simulation to actually find the number of circuit elements that switch when a vector is scanned in. However this procedure takes a very long execution time and is thus very expensive. Instead, to estimate the scan-in or the scan out power, we use the weighted transitions metric proposed by [Sankaralingam 00]. In their paper, they have found that the sum of average weighted scan-in transitions and the average weighted scan-out transitions is fairly closely correlated to the average number of circuit elements that make transitions in the CUT. The weighted transitions metric model can be explained as follows.

Consider the previous example of the scan-in vector 1001. As shown in Fig. 4.1, there are two transitions in the scan vector. While Transition 1 dissipates power at every cell in the scan chain while being scanned in, Transition 2 only dissipates power at the first scan cell. Thus when a test vector is being scanned in, the number of scan cell transitions caused by a particular transition in that vector would depend on the position of the transition in the scan vector. According to [Sankaralingam 00], the weight assigned to a transition is the difference between the size of the scan chain and the position in the vector in which the transition occurs. The number of weighted transitions is given by:

Weighted_Transitions = Σ (Size_of_Scan_Chain - Position_of_Transition)

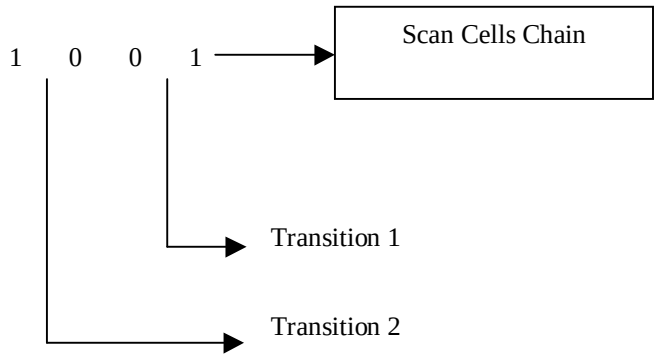

Figure 4.1: Transitions in example scan vector [Sankaralingam 00]

4.2.2 Estimation of Area overhead

In an algorithm that re-orders the scan chains to reduce the power dissipation, the main concern is whether the re-ordering increases the area overhead of the circuit. We have used two measures of the area overhead. One measure of the area overhead is the layout area, which measures the overall area of the chip. Another heuristic measure of area that we use in this chapter is the approximate area overhead, which is the sum of the Manhattan distance between two consecutive cells in the scan chain. This is an estimate of the stitching-wire length between consecutive cells in the scan chain, and is one of the biggest components to the area overhead [Makar 98]. Manhattan distance gives the estimate of the routing complexity – the longer the Manhattan distance, the greater is the routing complexity and the designer cannot compact area as much. So we have chosen the Manhattan distance as an estimate for approximate area overhead.

4.2.3 Minimum-transition Fill (MT-fill)

One important feature of our optimization algorithm is the fact that we do MT-fill of test vectors "on-the-fly", which we call dynamic MT-fill. Details of this method will be explained in Sec. 4.4. In this section, we give a background of MT-fill.

Consider a test vector matrix that has 0, 1 and X entries, where each row of the matrix corresponds to a test vector for the circuit. X is an unspecified value and can be filled with either 0 or 1. The conventional approach for filling the X's in the test cube is to do random fill (R-fill) in which the X's are randomly replaced by 0's or 1's. In R-fill, the idea is that it increases the chance that a single test cube would detect additional faults and hopefully the other test cubes would not be required and can be eliminated during reverse fault simulation. However, since we are considering power, which involves the number of weighted transitions in the test vector, it is best to consider Minimum Transition Fill (MT-fill). In MT-fill, a series of X entries in the test vector are filled with the same value as the first non-X entry on the right side of this series. This minimizes the number of transitions in the test vector when it is scanned in. For example, consider the test vector: 100**XX**010**X**1**X**0. This vector, after MT-fill, would become 100**00**010**1**1**0**0. If the test vector has a string of X bits that is not terminated by a non-X bit on the right side, then it should be filled by the bit value to the left of the sequence. For example: 1000001011**XX** should be 100000101**11** after MT-fill.

4.3 Scan Reorder Methodology for Minimizing Power and Area Overhead

In the conventional approach, a gate level netlist is generated after design synthesis. After that, scan insertion is done and then placement and routing is done. In our scan reorder methodology, the conventional process is modified as shown in Fig. 4.2.

In our methodology, after scan insertion we do placement to get the initial co-ordinates of the scan cells in the circuit. These co-ordinates are used for finding the best ordering of the scan cells using our proposed algorithm, which tries to do joint minimization of power and area overhead. The scan chain cells are re-connected according to this new ordering, after which placement and routing is done again to get the new co-ordinates of the re-ordered scan cells. The test vectors are also reordered according to the new ordering of the scan cells.

4.4. Algorithm for Ordering Scan Cells

As mentioned earlier, in our algorithm (shown in Fig. 4.4), we start with a test vector matrix that has 0, 1 and X entries, where each row of the matrix corresponds to a test vector for the circuit. Each test vector column corresponds to a scan cell in the scan chain, such that ordering the columns in the test vector matrix is equivalent to ordering the scan cells in the scan chain.

In order to find the best ordering of the scan cells in the scan chain, we try to jointly minimize power and area overhead and study a trade-off between the two. For that, we define a score function between two columns i and j of a test vector matrix:

$$Score(i,j) = Distance(i,j) + \lambda * Power(i,j)$$

where λ is the trade-off parameter between distance and power. *Distance(i,j)* is measured by the Manhattan distance between the scan cells corresponding to

columns *i* and *j* (as explained in Sec. 4.2.2), while *Power(i,j)* is measured by weighted transitions (as explained in Sec. 4.2.1).

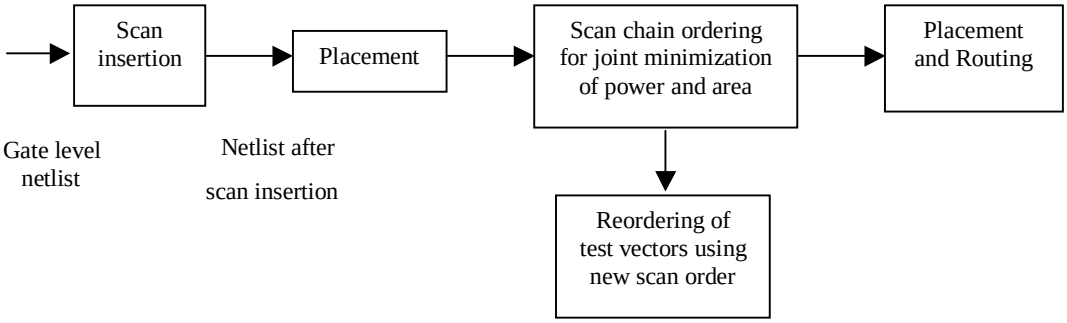

Figure 4.2: Scan Cell Reordering Methodology

For a given value of λ, the algorithm starts by choosing the 2 columns in the test matrix that have the minimum score between them, in the function findBestSeeds. These two columns are placed on the rightmost part of the test matrix, as "seeds". This is because, considering that the test vectors are inserted into the scan chain from the left, the transitions between the two columns on the rightmost side of the test matrix would have the maximum weight (as explained in Sec. 4.1). So, we want to greedily seed the re-ordering process by assigning the two columns with minimum score between them as the rightmost columns in the re-ordered matrix. After that, the column before the rightmost one is MT-filled with respect to the rightmost column. We refer to this process of doing MT-fill “on-the-fly” as dynamic MT-fill.

We illustrate this technique of dynamic MT-fill with the following example. After selecting the best two columns for seeding, the algorithm places them as the rightmost two columns of the test matrix. After this step let the example test vector matrix look like Fig. 4.3 (a). The algorithm does dynamic MT-fill at this

stage. The current column under consideration (the second column from the right) gets changed by specifying the X's such that the number of transitions between the current column and the rightmost column is minimized. The resulting matrix is shown in Fig. 4.3 (b).

In the main loop of the function findBestOrdering, we consider every column *i* in the matrix from right to left (due to the weighting scheme), starting from the column before the rightmost one. The function greedyColumnToSwap in the algorithm finds the column *j* in the matrix, to the left of *i*, which has the minimum score with column *i*. It then swaps column *i-1* with column *j*. After swapping, the new column *i-1* is dynamically MT-filled with respect to column *i*.

The dynamic MT-fill done at every step of the re-ordering process is a novel technique, which greatly contributes to the improved performance of our algorithm. Instead of doing the dynamic MT-fill at every step, if we had performed the MT-fill on the total test vector matrix at the beginning of the algorithm, then we would have lost degrees of freedom in choosing the best values with which to fill up the X values in the test vectors in order to minimize the number of transitions. If, on the other hand, we had chosen to do a MT-fill at the end of the algorithm, then we would have too many options for selecting the best column at every step of the algorithm. Dynamic MT-fill gives a good compromise between these two extremes, by selecting the X values in the current column so as to minimize the number of transitions at every step of the algorithm, while at the same time keeping enough degrees of freedom for selecting the best columns in the future steps.

At every step of the algorithm, the overall score (power and distance) between the test matrix columns is greedily minimized. At the end of the swaps, the new column ordering is output. This process is repeated for all values of the trade-off parameter λ that are specified by the designer.

1 1 2 0	1 1 **0** 0
2 1 2 0	2 1 **0** 0
0 1 0 0	0 1 **0** 0
1 0 2 1	1 0 **1** 1
(a)	(b)

Figure 4.3: Dynamic MT-fill of column 3 w.r.t. column 4

4.5 Experimental Results

The experimental methodology is as follows:

1. For a given circuit, insert the scan chain into it.
2. Run a placement and routing tool to get the locations of the scan cells and the total area overhead of the circuit.
3. Using the test-vectors and the locations of the cells, run the greedy algorithm to get the (local) optimal ordering of the cells for different values of the scaling parameter λ.
4. From Step 3, we get the approximate area overhead and estimated power of the circuit corresponding to the ordering for a particular value of λ. These values are used to plot the power and the approximate area overhead for each value of λ.
5. After looking at the plots in Step 4, we choose a particular value of λ that gives a good trade-off between power and area overhead. For this value of λ, we stitch the scan-chain according to the ordering found in Step 3. Running an area-measurement tool gives the layout area for this λ.

We performed experiments on the following circuits from the ISCAS-89 benchmark suite [Brglez 89]: s5378, s9234, s13207, s15850 and s38417. We used the wolfe tool [Sechen 85] in OctTools for routing, placement, and calculating the

area overhead of the circuit after placement. The results on the five benchmark circuits are shown in Figs. 4.5-4.7. In each figure, the initial estimated area of the circuit refers to the approximate area overhead of the circuit with the default ordering of the scan cells. The approximate area overhead is calculated as the sum of the lengths of the wires connecting the scan cells, i.e., the Manhattan distance between scan cells (as explained in Sec. 4.2). The initial estimated power is the power estimated from the number of transitions in the test vectors with the default ordering. For each value of λ, our algorithm finds the (local) optimal ordering by minimizing the combined power and area overhead metric. For each λ, the final estimated area is the approximate area overhead of the circuit after layout and placement, with the scan cells being ordered according to the optimal ordering, while the final estimated power is the power estimated from the number of transitions in the test-vector matrix after the ordering of the scan cells.

As can be seen from the graphs of all the circuits, increasing the value of λ increases the final approximate area overhead and decreases the final estimated power, after the scan chain has been re-ordered using the ordering output by the algorithm. For all of our circuits, the designer can choose the value of λ to trade-off the savings in power with increase in approximate area overhead by looking at the graphs. For each circuit, we chose a particular trade-off value of λ. For that value of λ, the percentage reduction of the actual layout area overhead of the circuit after scan-chain re-ordering was calculated, by running a placement and routing tool on the circuit with the reordered scan cells.

As can be seen from the graphs, our algorithm achieves very good reduction in estimated power and approximate area overhead for all the benchmark circuits. For the trade-off value of λ chosen by us corresponding to each circuit, the percentage reduction in estimated power and actual layout area is shown in Table 4.1.

```
Input:- TestMatrix[rowSize,columnSize] // Test vector matrix
        - LocationsVector[columnSize]   // Contains (X,Y) co-ordinates of the scan cells
        - LambdaVector[numLambdas]   // Contains diff. values of parameter lambda

Output: Best column ordering for each lambda value

findBestOrdering() {
  for each lambda in LambdaVector {
    findBestSeed(lambda);
    Dynamic MT-fill column (columnSize-1) with respect to column columnSize;
    for (i=columnSize-1; i>=2; i--) {
      bestColumn = greedyBestColumnToSwap(i,lambda);
      Swap column bestColumn with column (i-1);
      Dynamic MT-fill column (i-1) w.r.t. column i;
    }
    Output the column ordering
  }
}

findBestSeed(lambda) {
  Find the pair of columns [a,b] from TestMatrix that have minimum score(a,b,lambda)
  Swap column pair (a,b) with the two rightmost columns of TestMatrix
}

greedyBestColumnToSwap(i,lambda) {
  Find the column j in the TestMatrix that has the minimum score(i,j,lambda)
}

score(i,j,lambda) {
  Between columns i and j in TestMatrix,  compute
[wireLength(i,j)+lambda*transitions(i,j)]
}

wireLength(i,j) {
  Compute Manhattan distance between cell i and cell j, using LocationsVector
}

transitions(i,j) {
  Compute weighted 0->1 or 1->0 transitions between column i and column j of the
TestMatrix
}
```

Figure 4.4: Algorithm for Scan Cell Reordering

Table 4.1: Results showing the reduction in estimated power and layout area overhead for the chosen value of λ for the experimental benchmark circuits

Benchmark circuit	Size of scan chain	Chosen λ value	Reduction in estimated power	Reduction in actual layout area
s5378	164	100	48.89%	4.83%
s9234	211	100	47.17%	3.79%
s13207	638	500	70.20%	6.72%
s15850	534	500	58.29%	5.42%
s38417	1636	1000	61.50%	5.01%

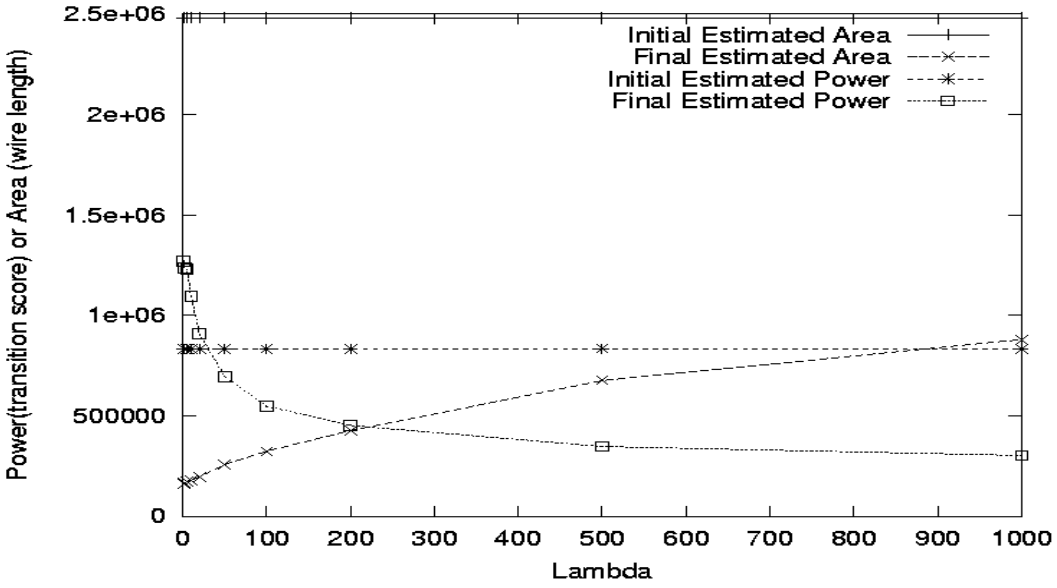

Figure 4.5: Results for s13207

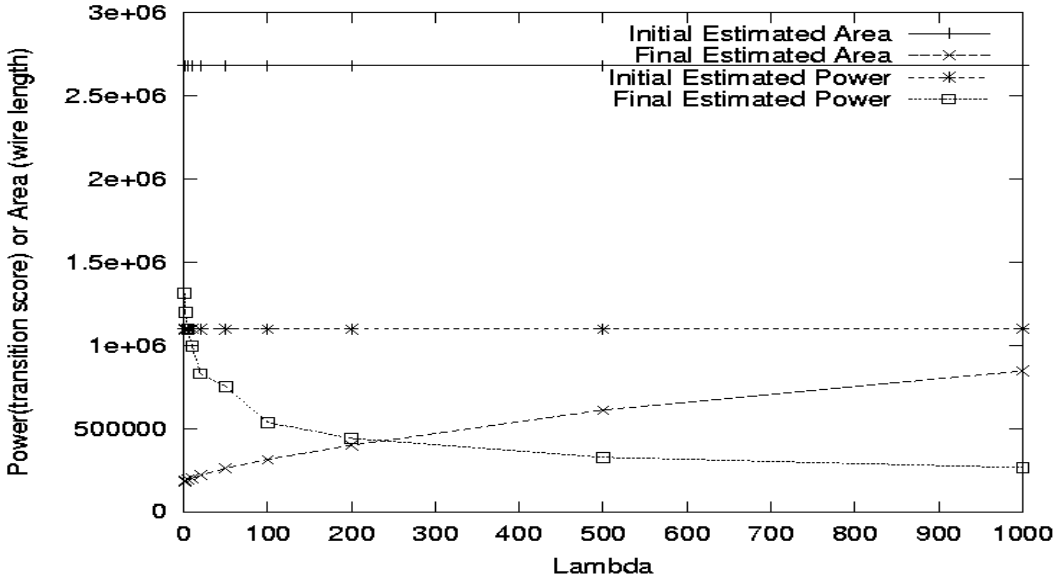

Figure 4.6: Results for s15850

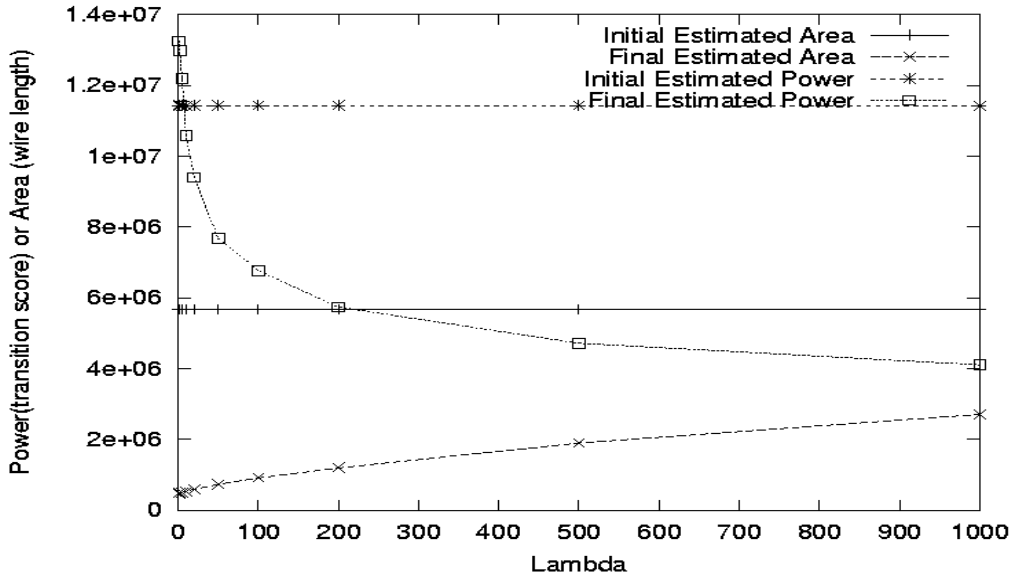

Figure 4.7: Results for s38417

Chapter 5

Low-Power Weighted Pseudo-Random BIST Using Special Scan Cells

In this chapter, another method for power reduction in offline testing is discussed. The proposed method is a technique for weighted pseudo-random built-in self-test (BIST) of VLSI circuits, which uses special scan cells and a new weight selection algorithm to achieve low power dissipation. It is based on weighted pseudo-random scan testing in which only 3 weight values are used – 2 fixed values (0 or 1) and 1 random value (0.5). A new weight selection algorithm is used to select a set of weights that achieves high fault coverage while reducing power. The idea is to minimize power by careful selection of the set of scan cells having fixed values (0 or 1) in order to reduce switching activity. To implement this in hardware, a new scan cell design is proposed that can do scan and capture in the normal mode as well as fixed-bit mode. The new scan cell hardware increases the area of a typical circuit by less than 4%, but reduces power by as much as 96%, as indicated in experiments performed on benchmark circuits [Ghosh 04a].

5.1 Overview

Built-in self-test (BIST) involves performing test pattern generation and output response analysis using on-chip circuitry. The most economical BIST techniques are based on pseudo-random pattern testing. There are two well-known problems with pseudo-random BIST: low fault coverage and high power dissipation. Low fault coverage arises due to the presence of random pattern

resistant (r.p.r.) faults [Eichelberger 83], which have low detection probabilities. Solutions to this problem involve either modifying the circuit-under-test (CUT) by inserting test points to increase the detection probabilities, or by modifying the test pattern generator so that it generates patterns that detect the r.p.r. faults. The problem of high power dissipation comes from the fact that pseudo-random patterns cause much greater switching activity in the CUT than what occurs during normal functional operation. This can result in overheating, as the chip package may only be capable of handling the power dissipation that occurs during functional operation. Moreover, for portable electronics where BIST is used out in the field, it is desirable that the BIST use a minimal amount of energy to preserve battery life.

The problem of low fault coverage for pseudo-random BIST has been studied for a long time and quite a number of solutions have been proposed. One of the most attractive involves adding weight logic to bias the pseudo-random patterns towards those that detect the r.p.r. faults. A number of weight selection algorithms have been proposed for finding a minimal number of weight sets to achieve a desired fault coverage [Brglez 89, Pomeranz 93].

The problem of reducing power dissipation during BIST is a more recent problem that has gained attention. Some interesting techniques have been proposed including the following: dual-speed linear feedback shift register (DS-LFSR) [Wang 97A], low transition random test pattern generation (LT-RTPG) [Wang 99], scan output holding [Gerstendorfer 99], test vector inhibiting [Girard 99A], circuit partitioning [Girard 99B], and modified clocking schemes [Girard 01].

An attractive approach that combines weighted pattern testing with a low power BIST was proposed by Wang in [Wang 02]. This approach can achieve high fault coverage by targeting random pattern resistant fault while still reducing power dissipation compared with conventional weight pattern testing. Three

things are combined in Wang's method: LT-RTPG, 3-valued weights, and scan re-ordering.

This chapter presents a new approach for combining weighted pattern testing with low power BIST. A different hardware scheme and weight selection methodology is used. Compared with Wang's approach [Wang 02], the proposed technique provides better scalability and power reduction.

5.2 Background

Pomeranz, *et al.*, [Pomeranz 93] introduced an algorithm for weighted pseudo-random testing, where the weight sets generated were *3-valued*, i.e., each weight value is 0, 0.5 or 1. We have modified this algorithm to generate weight sets that reduce the power dissipation during testing. Note that by *weight set*, we will refer to a vector in which each position can be 3-valued – such a weight set will be used to generate weighted pseudo-random test vectors.

The basic idea of the 3-valued weight set generation algorithm of Pomeranz, *et al.*, is to use a deterministic test set to find a small number of weight sets that can be used to generate weighted random test patterns for a circuit. The goal is to get significant fault coverage using a much smaller number of weights than the number of deterministic test vectors, thereby making the scheme useful for built-in self-test (BIST) due to its small area requirement. The test generation process starts with a set of purely random patterns to detect the easy-to-detect faults. After that, weight sets are computed by combining test vectors from the deterministic test vector set. Two test vectors *v1* and *v2* are combined using the following method: if *v1* has a value 1 (0) in a bit position, then the corresponding bit in the combined vector gets the value 1 (0) if *v2* has the same bit value 1 (0) or the value X at that position. If at any position *v1* has a bit value 1 (0) and *v2* has a different

bit value 0 (1) at that position, the corresponding position in the combined vector gets a value R. Note that X signifies an unspecified bit value, while R indicates a weight value of 0.5. For example, combining **XX**01100 and **X**00**X**001 would give **X**001**R**0**R**.

Once a weight set is created, the algorithm generates *N* weighted random patterns using the LFSR by fixing the scan cell positions corresponding to 0 or 1 weight values in the weight set and randomly changing scan cell positions having a weight value of 0.5. If any bit position in the weight set has a value of X, it is filled by minimum-transition (MT) fill [Sankaralingam 00]. In MT-fill, a series of X entries in a vector are filled with the same value as the first non-X (and non-R) entry on the right side of this series. This minimizes the number of transitions when such a vector is scanned into a scan chain. For example, consider that after the combination process described earlier, we get a weight set 1**X**0**R**0**R**10**X**1**X**0. This set, after MT-fill, would become 100**R**0**R**101100. If the set has a sequence of X bits that is not terminated by a non-X bit on the right side, then it is filled by the bit value to the left of the sequence. For example: 10**R**0**RR**1011**XX** will become 10**R**0**RR**1011**11** after MT-fill.

For each weighted random pattern, the algorithm performs fault simulation and removes the detected faults from the fault list. When no faults are detected for one run of *N* weighted random patterns for a weight set, it decreases the number of random values in the weight set *(K)* by 1, i.e., it fixes one more bit in the next weight set, in order to detect the harder-to-detect faults.

Our main motivation of using a 3-valued weight set generation algorithm is the fact that weight values of 0 or 1 can be fixed throughout generation of the weighted random patterns for a weight set and thereby reduce switching activity, and hence power. So, if we have hardware support to fix bit values in the scan cells, we can use a modified version of the 3-valued weight set generation algorithm tuned towards generating weight sets that fix bits corresponding to scan

INPUTS: ***F*** = target faults to be covered, ***C*** = minimum accepted fault coverage, ***K*** = initial number of random values in weight set, ***R*** = number of pure random patterns generated before weighted pattern generation, ***P*** = vector of estimated power saved by fixing each scan cell position at 0 or 1, ***N*** = number of weighted pseudo-random test patterns generated for each weight set, ***D*** = deterministic test set

OUTPUT: ***W*** = set of 3-value weight sets

WeightSetGeneration( ):

1. Let LFSR generate *R* pure random patterns, remove detected faults from *F*. Initialize set *W* as empty
2. Let originalNumberOfFaults = number of faults in *F*
3. While (number of current faults in *F*) > (1-***C***) * originalNumberOfFaults, repeat Step 4-6
4. Using *LowPowerWeightSelection( ),* generate weight set *w* having *K* random bits. Add *w* to set *W*
5. Generate *N* weighted random patterns using LFSR, by fixing scan cells positions with 0 or 1 weight value in *w*, and randomly changing scan cells having 0.5 values in *w*. For each random pattern, perform fault simulation and remove detected faults from *F*
6. If no fault is detected in Step 4, set *K* = *K* - 1
7. Return *W*

LowPowerWeightSelection(K):

1. For every fault in the set *F* of undetected faults, find the test vectors from *D* that cover the fault
2. Select a fault *f* that has minimum number of test vectors covering it. Among all test vectors that detect *f*, find the vector *t* that covers most number of faults
3. Mark vector *t*, and initialize weight set *w* = *t*
4. While (number of random bit positions in *w*) < *K*, repeat steps 5-6
5. Select the unmarked test vector *t'* from *D* that has the maximum *PowerSavedScore( )* with *w*
6. Set *w* = *FindIntersection(w,t').* Mark *t'*
7. Return *w*

PowerSavedScore(w,t'):
Initialize *saved*=0. For all positions *p* where *w* and *t' have* same bit value *b*, set *saved* = *saved* + (power saved by fixing position *p* at *b*, obtained from *P*). Return *saved*

FindIntersection(w,t'):
Initialize *intersection* to have R (i.e., 0.5) in all bit positions. For every position where both *w* and *t'* have same bit value *b* (0 or 1) or one has *b* and the other has X, set that position in *intersection* to *b*. Return *intersection*

Figure 5.1: Low power weight selection algorithm

cell positions giving maximum power savings. Accordingly, the combinational part of the circuit connected to the scan cells having the fixed bit values would not have any transitions during application of the weighted random patterns, resulting in reduced power consumption during testing. We modified the 3-valued weight set generation algorithm to make it power-sensitive, and modified the SFN cell design of AlShaibi, *et al.*, [AlShaibi 94] to create fixed-mode scan cells that enable us to implement our scheme in hardware.

5.3 Algorithm Description

This section gives the details of our low power weighted pseudo-random BIST algorithm.

5.3.1 Low-power weighted pseudo-random testing

Our algorithm takes the basic idea of the original 3-valued weight selection algorithm of Pomeranz, *et al.*, but makes an important change in the weight set generation step that reduces power consumption during testing.

In the original algorithm, a weight set is selected to maximize coverage. First, the faults are ordered in decreasing order of how hard-to-detect they are, estimated by the number of deterministic test vectors that cover the fault (the less the number of vectors covering a fault, the more it is hard-to-detect). Then, two test vectors are selected, each of which covers one hard-to-detect fault and also has high fault coverage. A weight set is created by taking the combination of these two vectors (details are shown in the *FindIntersection( )* function in Fig. 5.1 and explained in Sec. 5.2). The weight set is combined with other high fault coverage test vectors covering hard-to-detect faults, and the number of fixed bit

positions in the weight set decreases at each step (implying that the number of random bit positions increase). This is continued till the number of random bit positions in the weight set does not exceed the desired value *K*.

In our proposed low-power weight generation algorithm outlined in Fig. 5.1, we make this weight selection scheme power-sensitive. First, for each scan cell position in the circuit, we estimate the power saved if that position were to be fixed at a value of 0 or 1, details of which are outlined in Sec. 5.3.2. Then we select the highest-coverage vector for the hardest-to-detect fault, similar to the scheme of Pomeranz, *et al.* After that, instead of creating a weight set by choosing test vectors with high coverage, we select test vectors that will give maximum power saved (due to bit positions being fixed) when combined with the existing weight set. This ensures that during weight set creation, we fix those bits in the scan cell that will give us maximum power saved when we do weighted random pattern testing.

5.3.2 Estimating power saved by bit fixing

Wang, *et al.*, [Wang 02] has an elaborate method of estimating how much power is saved by fixing a particular circuit node to 0 or 1, by estimating the transition density for that node. However, calculating their metric for large circuits is time-intensive. We use a simulation-based approach that is fast and reasonably accurate for our purpose. We generate 100 random patterns and apply that to the circuit, calculating the average power dissipated per random pattern (*pR*) using a power simulation software. We then fix a scan cell position *i* to 1 (or 0) and calculate the average power dissipated by applying the same set of 100 random patterns, keeping scan cell *i* fixed at 1 (or 0) – the difference of this value and *pR* gives us an estimate of the power saved in the circuit by fixing the bit

position *i* to 1 (or 0), which we denote *p1* (or *p0*) for position *i*. The values of *p1* and *p0* are estimated for each scan cell position in the circuit, giving us an estimate of the power saved in the circuit by fixing the scan cell positions to 1 and 0 respectively. These estimates are stored in vector *P* in Fig. 5.1.

5.4 Hardware Details

In this section, we will present the design of a fixed-bit scan cell that is motivated by the SFN cell design of AlShaibi, *et al.* [AlShaibi 94]. We will demonstrate its correctness using simulation results and also analyze the area overhead of this new scan cell design. Note that Pomeranz, *et al.*, [Pomeranz 93] implemented bit fixing in their scheme by using a bit-fixing control logic between the output of normal scan cells and the combinational parts of the circuit. The problem with this approach is that the bit-fixing logic becomes quite complex for large circuits, incurring area overhead as well as delay overhead in the normal operation of the circuit. This prompted us to generalize the design of the scan cells to support bit fixing, so that our proposed approach scales well to large circuits.

5.4.1 Fixed-bit scan cell design

The SFN (Scan-Fixed-Normal) cell design of AlShaibi, *et al.*, is a scan cell capable of scanning-in values in the normal mode as well as the fixed mode. The normal mode operation is same as a typical scan cell. In the fixed mode, the output of the SFN cell that is connected to the combinational part of the circuit is held constant while values are scanned through the SFN cell in the scan chain, thereby reducing power consumption in the combinational circuit components connected to the SFN cell output.

A major improvement of our fixed-bit scan cell design is that apart from normal and fixed scan, it is also capable of performing the *capture* operation in both normal and fixed modes, something that AlShaibi, *et al.*, did not consider in their design. Their scheme was for test-for-clock architecture, whereas we consider the standard test-per-scan STUMPS architecture for BIST [Bardell 82]. Scan capture is an essential operation of a scan cell in the standard STUMPS scan-BIST architecture, since here the scan chain captures the response of the circuit after application of the test vector. We call our design the SFNC (Scan-Fixed-Normal-Capture) cell, details of which are shown in Fig. 5.2.

The SFNC has 6 inputs: apart from the usual *Data_Input (DI)*, *Scan_Input (SI)*, *Scan* signal and *Clock* signal, it had 2 other control signals – *Fixed_Mode* and *Config_Load*. The SFNC cell has 2 outputs: *Scan_Output (SO)* and *Data_Output (DO)*. *DO* is connected to the combinational part of the circuit, while *SO* is connected to the *SI* of the next scan cell in the chain. Note that a normal scan cell has a single output line, which is connected to both the combinational part of the circuit (like *DO*) as well as to the scan-in of the next scan cell in the chain (like *SO*).

In our SFNC cell, the *M* and *S* latches together make a traditional flip-flop. The *C* latch holds the information whether the cell is fixed ($C = 1$) or not ($C = 0$), and the *F* latch holds the fixed bit value. When the SFNC cell is configured to be operating in the fixed mode so that $C = 1$, the output mux connects the output of *F* to *DO*. So, during scan and capture operations in this fixed mode, values shifted into the scan cell are shifted out through *SO*, but the combinational circuit connected to the cell sees the fixed bit output of the *F* latch through *DO*. So, in the fixed mode, there is no power dissipation in the combinational circuit connected to the output of the SFNC cell, since the *DO* output is held constant. In the normal mode, when $C = 0$, the SFNC cell behaves like a normal scan cell as *DO* gets driven by *SO*.

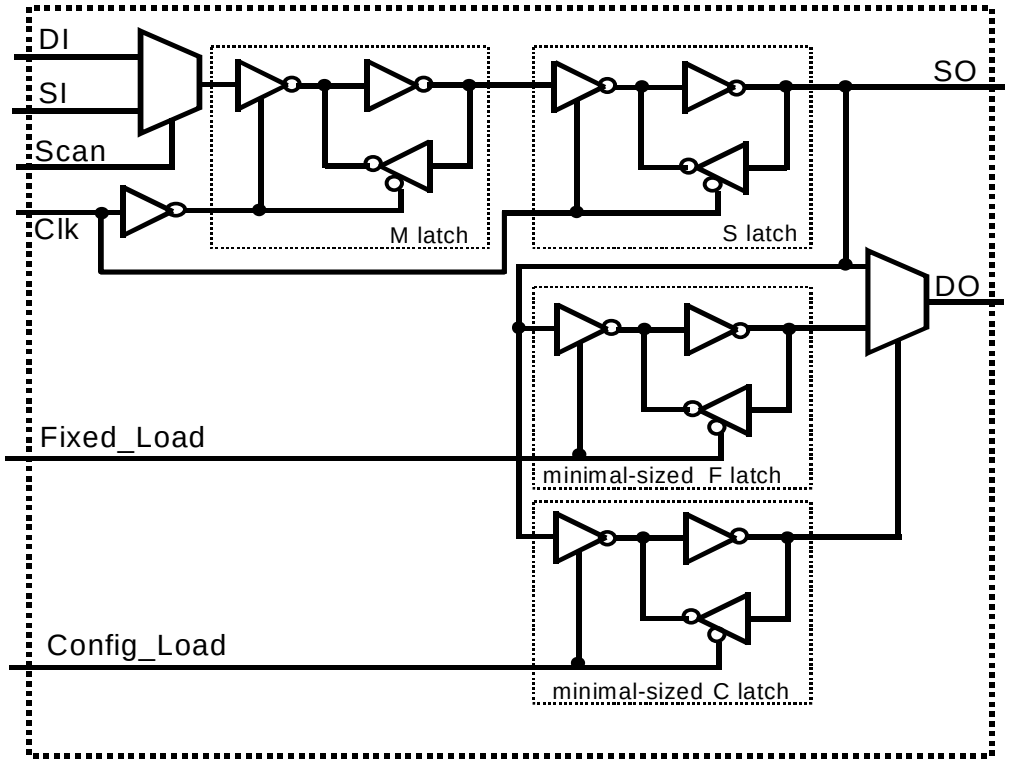

Figure 5.2: SFNC Scan Cell Design

The values in the *F* and *C* latches must be scanned in separately in two different *n*-cycle scan sequences. In the first *n*-cycle sequence, the actual weight set is scanned in, which ends with asserting *Fixed_Load* for one cycle. This captures the output of the *S* latch into the *F* latch. The next sequence loaded is the configuration vector, which configures the *C* latches in each SFNC cell: it ends with a *Config_Load* assertion for one cycle loading the value held in the *S* latch into *C*. The configuration vector that is scanned in has a 1 corresponding to weight set positions having a fixed bit (0/1), and 0 corresponding to a weight set position having a random bit (0.5). For example, for the weight set 1XX110XX01, the configuration pattern would be 1001110011. Once the fixed

values are loaded and the SFNC cells are configured, the LFSR is run to generate *N* pseudo-random test patterns. Each of these *N* random test patterns is weighted according to the weight set loaded into the scan chain.

At power-up of the circuit, the *C* latch in each SFNC cell needs to reset by the functional reset of the latches, preferably using asynchronous reset signals. This would make sure that the SFNC cell works in the normal mode at power-up. The *F* and *C* latches are minimum sized latches, i.e., their transistors are as small as possible, since these 2 latches drive small loads and are not timing-critical. The mux at the output would need to be large to drive the presented load. All latches in the design are level sensitive.

5.4.2 Estimation of area overhead

A standard scan cell has 2 typical latches, 1 mux, and 1 inverter, with 4 inputs and 1 output. Our SFNC scan cell design has 2 typical latches, 2 minimal-size latches, 2 muxes, and 1 inverter, with 6 inputs and 2 outputs. Let *X* be the typical latch area, *Y* be the inverter area, *Z* be the mux area and *X'* be the minimum latch area. So the percentage area increase of a SFNC cell over a standard cell would be $(2X' + Z) *100 / (2*X + Y + Z)$.

Considering 130nm technology, typical values of *X, Y, Z* and *X'* would be 30, 7, 15 and 9 units respectively. So, the SFNC scan cell would be about 40% more in size compared to the standard scan cell. For example, consider a chip where 40% of the area is occupied by memory and 60% by logic, of which about 15% is for flip-flops. If all the flip-flops were replaced by SFNC flip-flops in the chip, the total chip area would increase by only 3.6%.

Our proposed STUMPS architecture with SFNC cells is shown in Fig. 5.3. The overhead consists of an increase in the size of the scan cells (as explained above) and a ROM for storing the configuration bits and weight sets. Note that

other BIST schemes that detect random pattern resistant faults also generally require a similar ROM, e.g., for storing weight sets for a weighted pseudo-random scheme [Brglez 89], or for storing seed patterns in a LFSR re-seeding scheme [Krishna 01], etc.

Apart from generating the usual scan testing control signals, the control logic shown in Fig. 5.3 generates the extra control signals (e.g., *Config_Load*, *Fixed_Load*) for scanning in the configuration and weight sets. The mux at the input of the scan chains is needed to select either the next weight (or configuration) set from the ROM, or the next random pattern from the LFSR.

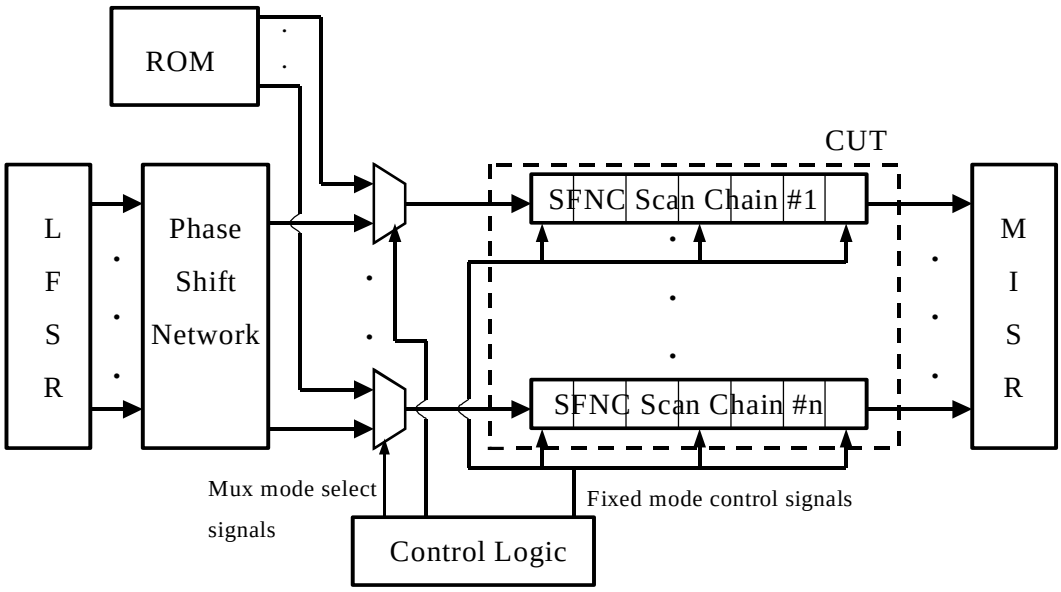

Figure 5.3: STUMPS architecture with SFNC cells

5.4.3 Simulation results

The correctness of the operation of the SFNC cell was verified by Verilog

simulation on a test circuit that has four SFNC cells (numbered 0-3) functionally connected as a shift register. The waveforms are displayed in Fig. 5.4. The

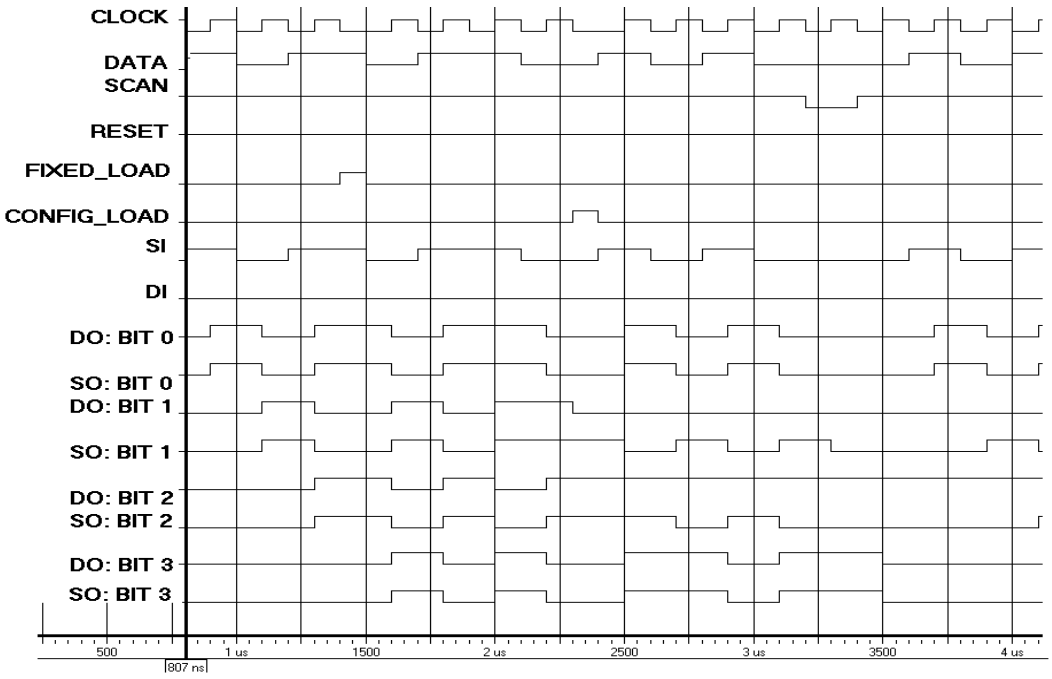

Figure 5.4: Waveform of test circuit

weight set is scanned in the first four cycles: *SFNC-1* is fixed to 0 and *SFNC-2* is fixed to 1. Subsequently, *Fixed_Mode* is asserted. In the next four cycles, the configuration vector is scanned in: it configures *SFNC-0* and *SFNC-3* as normal scan cells, and *SFNC-1* and *SFNC-2* as fixed scan cells, followed by the assertion of *Config_Load*. From this point on, note that the data out of *SFNC-1* and *SFNC-2* in Fig. 5.4 hold at their respective values, showing that the SFNC cell is performing correctly. We then do a scan in, capture and scan out – all the while the *DO* output of cells *SFNC-1* and *SFNC-2* remain unchanged and the *DO*

outputs of cells *SFNC-0* and *SFNC-3* change with their corresponding *SO* values, as desired.

5.5 Experimental Results

Experiments were performed on 4 standard ISCAS-89 benchmark circuits: s5378, s9234, s13207 and s15850. For each circuit, we used the ATALANTA toolkit [Lee 91] to generate the deterministic test vector set. We initially performed random test pattern generation to detect the easy-to-detect faults. In these experiments, an initial set of 1024 random patterns was generated. For designing weight sets to detect the remaining faults, we ran the *WeightSetGeneration* algorithm in Fig. 5.1, with parameters N = 256 (i.e., corresponding to each weight set we generated 256 weighted random test patterns), F = number of detectable faults in the circuit, C = 0.98 (i.e., minimum accepted fault coverage on all detectable faults is 98%) and K = 0.10 * number of scan cells in the circuit (i.e., a maximum of 10% of the weight values were random, implying at least 90% of the weight values were fixed in each weight set).

The power dissipation for each test set was estimated by counting the number of node transitions in the whole circuit and weighting each node transition by the number of fanouts at the node. As can be seen from the results in Table 5.1, we get greater than 98% fault coverage for detectable faults on all circuits. For each weight set, we have to store the actual weight pattern and the configuration pattern in a ROM, but the number of weight sets is an order of magnitude less than the number of deterministic test vectors. So, our scheme is very well suited for a BIST environment. Table 5.1 also shows the ROM size required to hold the

Table 5.1: Results on low-power weighted pseudo-random testing algorithm on ISCAS-89 benchmarks

Circuit Name	#Scan Cells	#Transitions with Algorithm in 0 & Normal Scan Cell	#Transitions with new Algorithm & SFNC Scan Cell	Fault Coverage	Size of Weight ROM (bits)	Power Reduction
s5378	214	7.11×10^8	2.49×10^7	99.58%	5K	96.49%
s9234	247	2.16×10^9	3.25×10^8	99.11%	25K	84.95%
s13207	611	4.65×10^{10}	6.85×10^8	98.45%	35K	95.85%
s15850	700	7.05×10^9	4.66×10^8	98.89%	17K	93.39%

weight sets for each benchmark circuit. Each weight set requires 2 bits per scan cell (for the weight set and configuration sequence). The numbers in Table 5.1 assume no compression. Note, however, that the data stored in the ROM is highly compressible, so it is possible to reduce the size of the ROM considerably using an approach along the lines of what is used in [Jas 01]. The power reduction compared with the original weight set selection algorithm of Pomeranz, *et al.*, ranges between 84% to 96% on these benchmark circuits, showing that our scheme will be very effective in low-power BIST for power-critical applications.

Note that with comparable fault coverage, the power reductions we have obtained are more than Wang's approach [Wang 02], presumably because we have fixed bit positions more aggressively in our algorithm (at least 90% of the bit positions are fixed for each circuit). Moreover, our hardware design is more scalable for large circuits, since we don't have to design special-purpose decoding logic for weight generation for every new circuit, as is required for Wang's WR-BIST hardware [Wang 02].

Chapter 6

Conclusions and Future Work

This chapter gives a summary of the completed work and discusses possible future research directions.

6.1 Summary

The primary focus of this dissertation has been power reduction in online testing and offline testing of circuits. The first two chapters have described methods for power reduction in concurrent error detection of systems with online testing. Chapter 2 described a method for low-power error-correcting code checkers for memory circuits, while Chapter 3 outlined a pre-synthesis technique for designing low-power parity prediction codes. The next two chapters of the dissertation discussed power issues in offline testing. Chapter 4 described a technique of minimizing power dissipation in a scan-tested circuit by re-ordering scan cells. Chapter 5 focused on BIST, a popular methodology for offline circuit testing, and a new method was proposed for combining power-conscious weighted pseudo-random pattern testing with a new low power scan-based BIST hardware.

6.2 Future Work

While this dissertation has discussed methods of reducing power in several important problems of online testing and offline testing for circuits, there are several other problems where power reduction is a critical issue. In this section, we give an outline of the further research problems that are of interest.

6.2.1 Power reduction in TMR circuits

In many critical hardware applications, dependability of the system and integrity of the data are very important. However, with the increase in density and reduction in size of integrated circuits, along with the lowering of voltage levels and reduction of noise margins, systems have become more susceptible to transient and intermittent faults. One important type of fault-tolerant system design is Tri-modular Redundancy (TMR), where three copies of a circuit are made and their output is voted, so that a fault in a single component can be masked by correct outputs from the other two components.

An important concern with such systems is the high power dissipation, due to triple replication of the circuit. One way to reduce power in TMR circuits would be to identify which components have high fault susceptibility and only replicate these components, instead of replicating the whole circuit. This will enable reduction of the power consumption of TMR circuits for battery-critical applications, while at the same time not affecting the fault coverage drastically.

6.2.2 Power reduction in Delay Testing

Delay testing is used to detect delay faults, i.e., timing defects in the circuit, in order to make sure that the design of the circuit meets the timing specifications. Delay testing has become more important with advances in transistor integration technology and the need for higher clock speeds, since delay faults can seriously degrade the timing performance of high-speed IC designs. So, there has been significant recent research in delay testing.

In low power devices, power reduction during testing is becoming more and more critical. However, not much research has been done so far in reducing power for delay testing. This would be a very interesting area of future research.

The main idea is as follows. To activate and detect a delay fault, a signal transition has to be propagated through the circuit. This requires the application of 2 test patterns to the circuit inputs to detect each delay fault. The transitions that are introduced in the delay path during application of the test pattern pair are important for detection of the timing defect. But, along with that other transitions are also created in the circuit, which cause power dissipation. In order to reduce power during delay testing, one will have to identify which transitions are non-critical during delay testing and minimize them, while at the same time keep the important transitions that are necessary for detecting the delay faults under consideration.

Bibliography

[Almukhaizim 04] Almukhaizim, S., P. Drineas, and Y. Makris, "Cost-Drive Selection of Parity Trees", *Proc. of the IEEE VLSI Test Symposium (VTS)*, pp. 319-324, 2004.

[AlShaibi 94] AlShaibi, M.F., and C. R. Kime, "Fixed-biased Pseudo-random Built in Self-test for Random Pattern Resistant Circuits", *Proceedings of International Test Conerence.*, pp. 929-938, 1994.

[Bardell 82] Bardell, P.H., and W.H. McAnney, "Self-Testing of Multichip Logic Modules", *Proceedings of International Test Conerence*, pp. 200-204, 1982.

[Bolchini 97] Bolchini, C., F. Salice, and D. Sciuto, "A novel methodology for designing TSC networks based on the parity bit code", *Proc. of the European Design and Test Conference (ED&TC-97)*, pp. 440 – 444, 1997.

[Bonhomme 01] Bonhomme, Y., "A Gated Clock Scheme for Low Power Scan Testing of Logic Ics or Embedded Cores", *Proceedings of Asian Test Symposium*, pp. 253-258, 2001.

[Brglez 89] Brglez, F., C. Gloster, and G. Kedem, "Hardware-Based Weighted Random Pattern Generation for Boundary Scan", *Proceedings of International Test Conference*, pp. 264-274, 1989.

[Burger 96] Burger, D., T.M. Austin, and S. Bennett, "Evaluating Future Microprocessors: the SimpleScalar Tool Set", *TR-1308*, Univ. of Wisconsin-Madison, CS Dept., July 1996.

[Chen 84] Chen, C.L., and M.Y. Hsiao, "Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the-Art Review", *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 124-134, March 1984.

[Chow 94] Chow, R.M., K.K Saluja, and V.D. Agrawal, "Power Constraint Scheduling of Tests", *Proceedings of International Test Conference*, pp. 271-274, 1994.

[Dabholkar 98] Dabholkar, V., S. Chakravarty, I. Pomeranz, and S.M. Reddy, "Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application", *IEEE Transactions on Computer-Aided*

Design, Vol.17, No. 12, pp. 1325-1333, December, 1998.

[De 94] De, K., C. Natarajan, D. Nair, and P. Banerjee, "RSYN: A system for automated synthesis of reliable multilevel circuits," *IEEE Trans. on VLSI Systems*, vol. 2, pp. 186 – 195, June 1994.

[Dell 97] Dell, T.J., "A White Paper on the Benefits of Chipkill-Correct ECC for PC Server Main Memory", *IBM Microelectronics Division*, November, 1997.

[Devadas 95] Devadas, S., and S. Malik, "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits", *Proceedings of Design Automation Conference*, pp. 242-247, 1995.

[Eichelberger 83] Eichelberger, E.B., and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test", In *IBM Journal of Research and Development* 27(3), pp. 265-272, 1983.

[Favalli 97] Favalli, K., and C. Metra, "Design of Low-Power CMOS Two-Rail Checkers", *Journal of Microelectronics Systems Integration*, vol. 5, no. 2, pp. 101-110, 1997.

[Gerstendörfer 99] Gerstendörfer, S., and H.-J. Wunderlich, "Minimized Power Consumption for Scan-Based BIST", *Proceedings of International Test Conference*, pp. 77-84, 1999.

[Ghosh 03] Ghosh, S., S. Basu, and N. A. Touba, "Joint Minimization of Power and Area in Scan Testing by Scan Cell Reordering", *Proceedings of the International Symposium on VLSI (IS-VLSI)*, 2003.

[Ghosh 04a] Ghosh S., E. MacDonald, S. Basu, and N. A. Touba, "Low-Power Weighted Pseudo-Random BIST Using Special Scan Cells", *Proceedings of the Great Lakes Symposium on VLSI (GLS-VLSI)*, 2004.

[Ghosh 04b] Ghosh S., S. Basu and N. A. Touba , "Reducing Power Consumption in Memory ECC Checkers", *Proceedings of the IEEE International Test Conference (ITC)*, 2004.

[Ghosh 05] Ghosh S., S. Basu and N. A. Touba, "Synthesis of Low Power Parity Prediction Circuits" *Proceedings of the VLSI Test Symposium (VTS)*, 2005.

[Girard 99A] Girard, P., L. Guiller, C. Landrault, S. Pravossoudovitch, J. Figueras, S. Manich, P. Teixeira, and M. Santos, "Low-Energy BIST design: Impact of the LFSR TPG parameters on the weighted switching", *Proceedings of ISCAS'99: International Symposium on Circuits and Systems*, 1999.

[Girard 99B] Girard, P., L. Guiller, C. Landrault, and S. Pravossoudovitch, "Circuit Partitioning for Low Power BIST Design with Minimized Peak Power Consumption", *Proceedings of Asian Test Symposium*, pp. 89-94, 1999.

[Girard 99C] Girard, P., L. Guiller, C. Landrault, and S. Pravossoudovitch, "A Test Vector Inhibiting Technique for Low Energy BIST Design", *Proceedings of VLSI Test Symposium*, pp. 407-412, 1999.

[Girard 01] Girard, P., L. Guiller, C. Landrault, S. Pravossoudovitch, and H.J. Wunderlich, "A Modified Clock Scheme for a Low Power BIST Test Pattern Generator", *Proceedings of VLSI Test Symposium*, pp. 306 –311, 2001.

[Girard 02] Girard, P., "Survey of Low-power Testing of VLSI Circuits", *IEEE Design and Test of Computers*, Vol 19, No. 3, pp. 82 – 92, 2002.

[Goessel 93] Goessel, M., and S. Graf, *Error Detection Circuits*, London, NY: McGraw-Hill, 1993.

[Gray 00] Gray, K., "Adding Error-Correcting Circuitry to ASIC Memory", *IEEE Spectrum*, pp. 55-60, Apr. 2000.

[Hamming 50] Hamming, R.W., "Error Detecting and Error Correcting Codes", *Bell System Technical Journal*, 29, 147, 1950.

[Hertwig 98] Hertwig, A., and H.J. Wunderlich, "Low Power Serial Built-In Self Test", *Proceedings of European Test Workshop*, pp. 49-53, 1998.

[Hochbaum 85] Hochbaum, D., and D. Shmoys, "A best possible heuristic for the *k*-center problem", *Mathematics of Operations Research*, Vol. 10, No. 2, pp. 180 – 184, 1985.

[Holland 75] Holland, J.H., "Adaption in Natural and Artificial Systems", *University of Michigan Press*, Ann Arbor (USA), 1975.

[Hsiao 70] Hsiao, M.Y., "A Class of Optimal Minimum Odd-weight-column SEC-DED Codes", *IBM Journal of Research and Development*, vol. 14, no. 4, pp.

395-401, July 1970.

[Huang 99] Huang, T.C. and K.J. Lee, "An Input Control Technique for Power Reduction in Scan Circuits During Test Application", *Proceedings of the Asian Test Symposium*, pp 315-320, 1999.

[Jas 01] Jas, A., C.V. Krishna, and N.A. Touba, "Hybrid BIST Based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning Scheme", *Proceedings of VLSI Test Symposium*, pp. 2-8, 2001.

[Kirkpatrick 83] Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi Jr., "Optimization by Simulated Annealing", *Science*, pp. 671-680, May, 1983.

[Kleihorst 01] Kleihorst, R., and N. Benschop, "Fault Tolerant ICs by Area-Optimized Error Correction Codes", *Proc. of International On-Line Testing Workshop*, pp. 143, 2001.

[Krishna 01] Krishna, C.V., A. Jas, and N. Touba, "Test Vector Encoding Using Partial LFSR Reseeding", *Proceedings of International Test Conference*, pp. 885-893, 2001.

[Lee 91] Lee, H.K., and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits", *Technical Report No. 12-93*, Department of Electrical Engineering, Virginia Polytechnic Institute and State University, 1991.

[Lee 97] Lee, C., M. Potkonjak, and W.H. Mangione-Smith, "MediaBench: A Tool for Evaluating Multimedia and Communications Systems", *Proc. Of Micro 30*, 1997.

[Lin 96] Lin, K.H, C.S. Chen, and T.T. Hwang, "Layout-driven Chaining of Scan Flip-Flops", *IEE Proceedings of Computers and Digital Technology*, Vol. 143, No. 6, November 1996.

[Makar 98] Makar, S., "A Layout-Based Approach for Ordering Scan Chain Flip-Flops", *Proceedings of International Test Conference*, pp. 341-347, 1998.

[McCluskey 86] McCluskey, E.J., "Logic Design Principles", Prentice-Hall, NJ, 1986.

[Mohanram 02] Mohanram, K., and N.A. Touba, "Input Ordering in Concurrent Checkers to Reduce Power Consumption", *Proc. of IEEE Symposium on Defect*

and Fault Tolerance, pp. 87-95, 2002.

[Mohanram 03] Mohanram, K., and N.A. Touba, "Partial Error Masking to Reduce Soft Error Failure Rate in Logic Circuits", *Proc. of IEEE Symposium on Defect and Fault Tolerance,* pp. 433 – 440, 2003.

[Pomeranz 93] Pomeranz, I., and S. M. Reddy, "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set", *IEEE Transactions on Computer-Aided Design*, pp. 1050-1058, July 1993.

[Pouya 00] Pouya, B. and A. Crouch, "Optimization Trade-offs for Vector Volume and Test Power", *Proceedings of International Test Conference*, pp. 873-881, 2000.

[Rossi 02] Rossi, D., V.E.S. van Dijk, R.P Kleihorst, A.K. Nieuwland, and C. Metra, "Coding Scheme for Low Energy Consumption Fault-Tolerant Bus," *Proceedings of International On-Line Testing Workshop*, pp. 8-12, 2002.

[Rossi 03] Rossi, D., V.E.S. van Dijk, R.P Kleihorst, A.K. Nieuwland, and C. Metra, "Power Consumption of Fault Tolerant Codes: the Active Elements," *Proceedings of International On-Line Testing Symposium*, pp. 61-67, 2003.

[Sankaralingam 00] Sankaralingam, R., R. Oruganti and N. A. Touba, "Static Compaction Techniques to Control Scan Vector Power Dissipation", *Proceedings of VLSI Test Symposium*, pp. 35-42, 2000.

[Sankaralingam 01] Sankaralingam, R., B. Pouya, and N. A. Touba, "Reducing Power Dissipation During Test Using Scan Chain Disable", *Proceedings of VLSI Test Symposium*, pp. 319-324, 2001.

[Saposhnikov 96] Saposhnikov, Vl.V., A. Dmitriev, M. Goessel, V.V. Saposhnikov, "Self-dual parity checking - a new method for on-line testing", *Proc. of the IEEE VLSI Test Symposium (VTS)*, pp. 162 – 168, 1996.

[Sechen 85] Sechen, C., and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package", *IEEE Journal of Solid State Circuits*, vol. 20, pp. 510-522, 1985.

[Sentovich 92] Sentovich, E. M., K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, P.R. Stephan, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," *Technical Report*

Memorandum No. UCB/ERL M92/41, University of California, Berkeley, 1992.

[Shivakumar 02] Shivakumar, P., M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic", *Proc. of the International Conference on Dependable Systems and Networks*, pp. 389 – 398, 2002.

[Sogomonjan 93] Sogomonjan, E.S., and M. Goessel, "Design of self-parity combinational circuits for self-testing and on-line detection"**,** *Proc. of the IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, pp. 239 – 246, 1993.

[Stephenson 00] Stephenson, M., J. Babb, and S.P. Amarasinghe, "Bidwidth analysis with application to silicon compilation", *Proceedings of Programming Language Design and Implementation*, pp. 108-120, 2000.

[Szu 87] Szu, H., and R. Hartley, "Fast Simulated Annealing", *Physics Letters A*, vol. 122, no. 3,4, pp. 157 – 162, 1987.

[Touba 97] Touba, N.A., and E.J. McCluskey, "Logic Synthesis of Multilevel Circuits with Concurrent Error Detection", *IEEE Trans. on Computer-Aided Design*, Vol. 16, No. 7, pp. 783 – 789, 1997.

[Wang 97A] Wang, S., and S.K. Gupta, "DS-LFSR: A New BIST TPG for Low Heat Dissipation", *Proceedings of International Test Conference*, pp. 848-857, 1997.

[Wang 97B] Wang, S., and S.K. Gupta, "ATPG for Heat Dissipation Minimization for Scan Testing", *Proceedings of Design Automation Conference*, pp. 614-619, 1997.

[Wang 99] Wang, S., and S.K. Gupta, "LT-RTPG: A New Test-Per-Scan BIST TPG for Low Heat Dissipation", *Proceedings of International Test Conference*, pp. 85-94, 1999.

[Wang 02] Wang, S., "Generation of Low Power Dissipation and High Fault Coverage Patterns for Scan-based BIST", *Proceedings of International Test Conference*, pp: 834-843, 2002.

[Whetzel 00] Whetzel, L., "Adapting Scan Architectures for Low Power Operation", *Proceedings of International Test Conference*, pp: 863-872, 2000.

[Xilinx 03] Xilinx Applications Note: CoolRunner-II CPLD, “Single Error Correction and Double Error Detection (SECDED) with CoolRunner-II CPLDs”, *XAPP383 (v1.1)*, August 1, 2003.

[Zorian 93] Zorian, Y., “A Distributed BIST Control System for Complex VLSI Devices”, *Proceedings of VLSI Test Symposium*, pp: 4-9, 1993.

Vita

Shalini Ghosh did her Bachelors in 1997 from the University of Calcutta, India, where she majored in Physics with Honors. During her undergraduate she won the All India Second Prize from Govt. of India for her dissertation on radioisotopes. After doing a year of research on Computational Physics at the Saha Institute of Nuclear Physics in Calcutta, India, she joined the University of California at Santa Cruz in 1998. There she did her MS in Computer Engineering, with a thesis on fault modeling in VLSI testing. She joined the University of Texas at Austin in 2000, where she is currently doing her PhD in Computer Engineering under Dr. Nur A. Touba. Her research interests are in the areas of power optimization for fault-tolerant and dependable systems, concurrent error detection, system reliability, VLSI testing, design for testability, fault modeling, and architecture validation. She was the runner-up for the "Best Paper Award" in GLSVLSI-2004. She has been an external reviewer for the Journal of Low Power Electronics in 2003, a reviewer for ACM Computing Reviews and has been a member of the "Women in Engineering Program" at UT, Austin since 2000. She has worked twice as a summer intern at Intel Corporation (Santa Clara and Austin) and once at Synopsys (Sunnyvale).

Permanent address: 12C/2 Selimpore Bye Lane,
Dhakuria, Calcutta - 700031,
West Bengal, India.

This dissertation was typed by the author.